ADAPTIVE, DISTRIBUTED DECISION SUPPORT USING GENETIC LEARNING: APPLICATION TO A PRODUCTION SCHEDULING ENVIRONMENT

By

SIDDHARTHA BHATTACHARYYA

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1993

ACKNOWLEDGEMENTS

This work would not have been accomplished without the support of many. I am indebted to my supervisory committee chairman Dr. Gary J. Koehler for his invaluable guidance and patient encouragement throughout my four years in the program. He instilled a deep interest in the subject and a feel for quality that I will carry for a long time. Thanks are also due to the members of my committee and others of the Decision and Information Sciences faculty for their support. Thanks and appreciation go to my colleagues in the program for the numerous conversations and many good times that have made these last four years enjoyable. A special note of thanks goes to Mr. Haldun Aytug for the long hours of fruitful discussions.

The support of IBM Corporation, through partial funding for this research, and providing the shop-floor data for the empirical work, is sincerely appreciated. Thanks go to Ms. Jane Snowdon of IBM, Boca Raton, for help in this regard.

Finally, and most importantly, I would like to thank my family for their boundless love and support; my parents without whom my career would not have materialized; my grandparents for early kindling of curiosity; my sister; and last, but not in any way the least, my wife, for unwavering encouragement and faith from the very beginning.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

ADAPTIVE, DISTRIBUTED DECISION SUPPORT USING GENETIC LEARNING: APPLICATION TO A PRODUCTION SCHEDULING ENVIRONMENT

By

SIDDHARTHA BHATTACHARYYA

August 1993

Chairman: Professor Gary J. Koehler
Major Department: Decision and Information Sciences

Scheduling is a crucial activity in a manufacturing system. With traditional approaches, applications of optimization techniques to a mathematical formulation of the problem have proven to be intractable even for many idealized situations; thus, a variety of Artificial Intelligence (AI) approaches have been proposed for effective production scheduling. These techniques too, however, have felt the limitations imposed by the lack of adequate means to acquire the requisite knowledge.

This knowledge acquisition bottleneck is further aggravated by the complexity of the production scheduling environment and the lack of reliable human experts from whom to glean the required knowledge--hence, the need for the development of machine learning schemes to aid in production scheduling.

Genetic algorithms (GAs) provide a stochastic search strategy based on principles of biological evolution and are noted to be specially suited for application to complex, poorly understood search spaces. A framework for utilizing genetic algorithm based learning in typical decentralized factory-floor decision making environments is presented. A high level knowledge representation scheme for modelling the production environment is developed, with facilities for genetic learning within this scheme. Experimental results from job shop simulations considering initial stages of a semiconductor manufacturing line demonstrate the feasibility of the designed approach and provide insights for future enhancements.

A second part of this research focusses on the theoretical analysis of genetic algorithms. Most theoretical studies on GAs consider binary encodings of the search space, and the fundamental principle of minimal alphabets suggests the optimality of binary over higher cardinality representations for genetic processing. A growing number of successful applications using higher level representations, however, present a potential conflict between theory and practice.

This research undertakes a generalization of a recent model of binary GAs, providing a detailed characterization of their search behavior, to higher cardinality alphabets. Walsh functions have been widely used in studying binary GAs, and a generalization of the Walsh matrix terms to consider higher cardinality representations is obtained. This, and other identities derived in the analysis provide useful results for further studies of nonbinary GAs.

CHAPTER 1

INTRODUCTION

The field of Artificial Intelligence (AI) is concerned with automated systems that exhibit intelligent behavior as typically displayed by humans. This machine emulation of human-like intelligence is sought to be utilized in the design of systems capable of performing tasks requiring cognitive abilities.

Attempting a definition of AI leads one to philosophical considerations. One may, however, from an applications perspective, broadly consider it as "a process comprised of learning, reasoning, and the ability to manipulate symbols" (Kurzwiel, 1990, p. 16). The latter two (search and knowledge representation) have been the fundamental concerns of most AI work, and only recently has learning come to the forefront of AI research and been accorded the influence and importance that is its due. A more comprehensive definition comes, perhaps, through consideration of the different AI sub-disciplines and application areas. Based on these, AI work may be grouped along the following lines: languages and environments for AI, modelling human performance, natural language understanding and semantic modelling, planning and robotics, automated reasoning and theorem-proving, game playing, expert systems, and machine learning (Luger and Stubblefield, 1989).

Expert systems (ES) have, by far, had the greatest commercial impact -- so much so, that AI today has almost become synonymous with the words expert systems. ES grew out of the recognized importance of domain knowledge for intelligent problem solving. Early ES used a condition-action rule representation with a backward or forward chaining inferencing mechanism. Today, such knowledge based systems have access to a wider spectrum of representation and reasoning schemes.

In spite of their demonstrated success and power in application specific projects, ES can be said to have failed to live up to all their early expectations. This is primarily attributable to the significant difficulties encountered in the acquisition of the required knowledge -- the now famous Fiegenbaum bottleneck. Traditional knowledge-engineer mediated techniques of knowledge acquisition through interviewing domain experts have been found to be largely infeasible. A growing body of work in machine learning now addresses this task of automated knowledge acquisition (Shalin, 1990). A number of machine learning paradigms are currently under active study. In this research we focus on one such method: genetic algorithm based learning.

The development and awareness of AI tools and techniques have also lead to their increasing use across diverse disciplines. One such area that has seen the widespread incorporation of knowledge-based techniques in recent years is production scheduling.

The scheduling problem, in general, has a wide range of application areas, that include production and project scheduling, mass transit scheduling, hydro-power scheduling, and manpower scheduling of shift-work (Noronha and Sarma, 1990). Traditional Management Science/Operations Research (OR) approaches have had limited impact in addressing the scheduling requirements of real world applications. In classical sequencing and scheduling theory, optimization algorithms are applied to problems that have been cast into a mathematical programming framework. The problems here quickly become intractable due to the combinatorial explosion of the search space, even when considering idealized problem situations. This difficulty is further aggravated by the lack of a means to incorporate domain specific knowledge for reducing the search. The mathematical formulation does not allow consideration of the full range of factory floor attributes and constraints of importance to a scheduler, and the single performance objectives considered also do not reflect the different tradeoffs required in real life scheduling. In a review of production scheduling, Graves (1981) notes the need "not only for better scheduling algorithms, but also for more realistic models of the scheduling setting and for increased understanding of the dynamics inherent in the scheduling environment" (p. 663). Since then, a variety of AI techniques have been applied to production scheduling. As observed in Rodammer and White (1988, p.841), "the development of complexity theory and maturation of artificial intelligence have begun to redirect the body of scheduling research."

Tracing back early AI based scheduling systems to Gere's work in 1962, Steffen reports that "research on AI based scheduling systems has tended to reflect the mainstream AI research of the time during which it was conducted" (Steffen, 1986, p.395). In the same paper, he asserts, however, that "while the goal of AI research is to simulate human intelligence with computers, when the methodologies developed from this research are applied to engineering problems, the goals are typically different . . . the goals are economic, not behavioral" (Steffen, 1986, p.397).

Knowledge based approaches to scheduling allow a more natural modelling of the environment and, as against most of their OR based counterparts, have also focussed on realistic problems. While most work in knowledge based scheduling is in the form of experimental prototypes, a number of systems are now operational in the factory floor and have demonstrated significant performance enhancements through their use (Fiegenbaum et al., 1988; Smith, 1991). A survey of knowledge based approaches to production scheduling is presented in Chapter 2.

As with ES work in other areas, researchers in knowledge based scheduling have also recognized the need for automated knowledge acquisition facilities. Conventional knowledge engineering techniques of interviewing domain experts and protocol analysis are specially ineffective here, due to the dynamic nature of the environment and human inadequacies in dealing with the complexity of the problem. It is contended that recognized and reliable experts just do not exist in many production scheduling environments (Blessing and Watford, 1987; Thesen and Lei, 1986; Yih, 1990a). The incorporation of a learning mechanism is thus now seen as crucial for the building of effective knowledge based scheduling systems.

This research is directed at the development of such techniques for the automated acquisition of knowledge required for production scheduling. Genetic Algorithm (GA) based learning is utilized, and a conventional job-shop scheduling environment considered.

More specifically, this research focusses on the design of an adaptive decision support environment utilizing genetic algorithm based learning. Recognizing the inherently decentralized nature of the production environment, the design seeks to facilitate coordinated decision making among distributed shop floor agents responsible for the scheduling of parts in a manufacturing facility. The incorporation of a genetic algorithm based machine learning paradigm provides an automated tool for the acquisition of the requisite decision making knowledge. It also provides the means for a continuous and incremental refinement of this knowledge, in keeping with dynamic scheduling activities and an evolving external environment.

The study of genetic algorithms, beginning with Holland's seminal work (1975), has progressed independently of the more behaviorally oriented mainstream AI research. Also, in contrast to most AI work, GA research has focussed on a number of real world problems (Davis and Coombs, 1987; Goldberg, 1989a). GAs provide a powerful parallel search strategy based on evolutionary principles and natural genetics, and have been successfully applied as a machine learning paradigm (Classifier Systems; see, for example, DeJong, (1990)). GAs are specially suited for application to complex and poorly understood search spaces. As Grefenstette (1989) notes,

> one of the more promising areas for application of incremental, competitive procedures such as genetic algorithms is the control of complex processes . . . that are not well characterized or are subject to a large number of uncontrolled, highly variable or unmeasurable parameters. Genetic learning techniques appear to offer an opportunity to learn high performance control rules without having good models of the process being controlled. (Grefenstette, 1989, p. 183)

A number of researchers have reported success with GAs applied across a wide spectrum of problems, from process control (Goldberg, 1985), communication network design (Davis and Coombs, 1987), to learning models of consumer choice (Greene and Smith, 1987) and simulation of rational agents in socioeconomic contexts (Forrest and Mayer-Kress, 1991). Some recent research efforts have also considered their application to scheduling problems (see Section 3.2).

GAs traditionally use a simple bit string representation of the search space. This constitutes a major disadvantage in attempting their use for scheduling in the factory floor. The effective modelling of the shop floor environment calls for a much richer representation.

The GA literature contains a number of references to systems that use high level representations with genetic search. In Grefenstette's (1989) SAMUEL system, a population comprises of rules expressed with a condition and an action part. Here, the condition of a rule is a conjunction of attribute-value subsets, and the action specifies environmental effector manipulations. Special purpose recombination operators have been suggested. The author notes that genetic learning techniques can be fruitfully applied to rule representation schemes that are quite different from that used in traditional classifier systems, and which express conditions at a fairly natural level. Maza (1989) describes another system that uses genetic learning of high level rules for picking winners in horse racing, a domain with little agreed upon knowledge of well performing strategies.

This research undertakes the design of a high level representation scheme, capable of adequately modelling the manufacturing environment and which is also amenable to manipulation by genetic search operators. The knowledge representation scheme developed aims at incorporating the different factory floor attributes, heuristics that human schedulers (experts?) might have found useful, and standard dispatching rules from the sequencing and scheduling literature. It also allows consideration of algorithmic models that designate jobs for dispatch. The traditional genetic search operators of crossover and mutation need to be modified for use with this high level representation, and the formulation of meaningful operators is explored. The design of an effective inference strategy and credit assignment scheme is also addressed.

The system is implemented in an object-oriented programming environment using C++. The various factory floor entities including work-areas, servers, queues, dispatchers, and jobs are modelled as object classes. Performance criteria, knowledge representation primitives, knowledge bases, and the genetic learning operators are also implemented using objects and associated methods.

A second part of this research undertakes a theoretical analysis of genetic algorithms. GAs are complex dynamical systems, and most theoretical work on their behavior is based on schemata analysis (Goldberg, 1989a). A fundamental principle--the principle of minimal alphabets--indicates that low cardinality representation schemes facilitate higher schemata processing. Use of a binary representation has been shown to maximize the implicit parallelism inherent in genetic processing. In contradiction to this, however, a number of researchers emphasize that higher cardinality alphabets are better suited for practical applications, and have greater utility and intuitive appeal. They hold out numerous successful applications as proof of the power and feasibility of nonbinary coded GAs. Goldberg (1990) admits:

> The debate between practitioner and theoretician over this *paradox of real codings* has risen almost to the point of schism. Theoreticians have wondered why practitioners have paid so little attention to the theory, and practitioners have wondered why the theory seems so unable to come to terms with their findings. (Goldberg, 1990, p. 1)

Noting that binary representations are a primary reason for GAs not finding wider acceptance, Antonisse (1989) provides an alternative interpretation of schemata, which contravenes the minimal alphabet principle, and argues for the use of nonbinary discrete alphabets. Wright (1991) examines real coded GAs in terms of schemata analysis and notes advantages over a binary representation. Goldberg (1990) also studies high cardinality alphabets and proposes a theory which suggests that initial selection pressures reduce high cardinality alphabets to lower cardinality virtual alphabets, which subsequently undergo processing through genetic operators. He also examines deceptiveness in genetic search when using such higher cardinality representations.

Recent work by Vose, Liepins and Nix (Vose and Liepins, 1991; Nix and Vose, 1992) undertakes a more detailed characterization of genetic recombination and provides a means for a more precise analysis of the evolutionary behavior of GAs. Modelling recombination through crossover and mutation as a dispersion operator, and selection as a focussing operator, Vose and Liepins (1991) obtain a mathematical formalization of the punctuated equilibria phenomenon often observed in genetic search--alternating periods of rapid evolution and generations of relatively stable populations. Conditions for the stability of fixed points for these operators are also derived. Nix and Vose (1992) apply this Vose-Liepins framework to model GAs as Markov chains, and provide convergence results and conditions. This work is further extended in Vose (1993).

This Vose-Liepins model is also, however, based on a binary representation. This research undertakes a generalization of this model to higher cardinality alphabets. A discrete alphabet using a fixed length string representation is considered, together with the simple one-point crossover and traditional mutation operators.

A survey of knowledge based approaches to production scheduling is presented in Chapter 2. This is followed in Chapter 3 by a review of machine learning applications to the scheduling problem. Genetic algorithms and classifier systems are then considered in Chapter 4, together with an analysis of the Vose-Liepins theoretical framework. Chapter 5 contains a description of the distributed decision support framework developed, and also details the knowledge representation scheme and inference strategies implemented, and the genetic recombination operators and credit assignment technique employed. Some empirical results on the effectiveness of the developed facilities in the automated acquisition of scheduling strategies in a simulated job shop manufacturing environment are presented in Chapter 6. Theoretical results on the generalization of the Vose-Liepins model are presented in Chapter 7. Chapter 8 then summarizes results and outlines directions for future research.

CHAPTER 2
SCHEDULING AND KNOWLEDGE BASED APPROACHES

2.1 Overview

Scheduling is a crucial activity in a manufacturing system and can heavily impact shop-floor efficiency. It has been noted that "work-in-process spends only 1.5% of its time in a value added activity; the other 98.5% of the time is spent in queues, movement, machine setup, quality inspection, etc." (Mayer et al., 1987, p. 11), thus emphasizing the need for establishing efficient scheduling practices.

Given its importance to the manufacturing enterprise, scheduling has been an area of active research since the 1960s. Traditional OR based techniques have, however, proved inadequate in addressing real world scheduling requirements. OR methods have focussed on the development of analytical and algorithmic solutions for highly constrained and static problems (Graves, 1981). The mathematical formalisms used do not allow consideration of realistic shop floor configurations and uncertainties inherent in the problem, and also fail to take into account the high level priorities and goals involved in scheduling (Buxey, 1989). It has also been noted that "productivity is not achieved in practice via mathematical calculations but by engineering know-how" (Buxey, 1989, p. 29). The field has thus seen the incorporation of a variey of Artificial Intelligence (AI) techniques in recent years. This chapter examines such AI or knowledge based approaches to the production scheduling problem.

In Section 2.2 we describe the production scheduling problem and scheduling environments. In Section 2.3, noting the inadequacies of traditional solutions in practical scheduling situations, we consider opportunities provided by the incorporation of knowledge based techniques. Then, in Section 2.4, we summarize various knowledge based approaches to scheduling reported in the literature.

2.2 The Scheduling Environment

Production scheduling involves the efficient allocation of available production resources over time so as to best satisfy certain performance criteria (Graves 1981). Consider a set of jobs to be processed over a given set of facilities. A job requires a set of operations, each of which can be performed on the given facilities subject to certain technologically feasible sequencing constraints. Developing a schedule requires detailing a sequence of operations for a job on the given facilities, such that the job is completed.

The production scheduling literature differentiates between flow shop and job shop environments. In a flow shop it is assumed that all jobs are processed on the same set of facilities and have the same precedence ordering of its operations. The job shop generalizes this situation. Here different jobs may require different processing steps, and a job may also have alternative operation sequences.

A second distinction may be made in terms of an open or a closed shop. In an open shop, customer orders drive the production tasks. In a closed shop, the items to be produced are determined through inventory replenishment considerations.

One may further distinguish between static and dynamic scheduling. In the former, plans are generated using forecasted data over a given time horizon, whereas in the latter, decisions are made in real time, depending on current shop-floor conditions.

A flexible manufacturing system (FMS) is another production setup in increasing prominence today. An FMS is designed for use in the manufacture of a large and changing variety of products. It is characterized by versatile machines that can perform a number of operations, an automated material handler to transport items between machines, and typically, centralized computer control for online decision making. Machines performing similar tasks are organized into work areas, and alternate operation sequences are permitted on jobs. The system is assigned item batches of varying sizes for manufacture. FMS scheduling involves sequencing the loading of different batches into the material handler, routing of items to various work areas, and allocation of items to specific machines within a work area.

Scheduling objectives vary, and measuring the performance of developed schedules generally involves tradeoffs between holding inventory, cost of frequent production turnovers, meeting due dates, utilization of resources, etc. Common performance criteria considered in the literature are percentage of late jobs, tardiness, makespan and flowtime. Most theoretical studies have considered such single criterion objectives. Real world environments, however, have to deal with multiple, often conflicting performance objectives, as noted below:

> While management typically seeks to minimize cost and maximize utilization of high-ticket machines and resources, scheduling objectives also frequently include objectives directed toward minimizing operating stress. Examples include improving schedule stability, reducing confusion, and placating a

demanding customer. Related objectives actually can imply deliberate under-utilization of machines to reduce queues and inventories or to ensure reliability. A complete general scheduler would need to capture and balance a great variety of performance criteria. (Rodammer and White 1988, p. 843)

Based on the time horizon of decisions, a three level hierarchical view is often taken of planning/scheduling activities: (1) strategic production planning, where lower level goals are set in accordance with management objectives; (2) short term tactical planning, involving release scheduling in keeping with short term production goals; and (3) item movement considerations at the production facilities (Ammons et al. 1988, Chryssolouris et al. 1988b, Smith 1991). While the first two steps are predictive in nature, the third involves reacting to situations and rescheduling. In fact, much of scheduling is considered to be actually a problem of rescheduling (Rodammer and White 1988).

Smith (1988b and 1991) attributes the difficulty of production scheduling to the following characteristics of the environment: (1) need to adhere to a diverse set of constraints related to demand, production processes, and resource availability, (2) unpredictability, (3) presence of multiple decision makers and need to balance amongst conflicting objectives and preferences, and (4) lack of real time information analysis facilities for coordinated and timely decision making.

2.3 Why Knowledge Based Scheduling?

2.3.1 Traditional Approaches

In traditional OR based approaches to scheduling, the problem is first cast into a mathematical programming formalism and then algorithmically solved to optimize some objective. Obtaining such optimal or near optimal solutions has been shown to be NP hard, even for many idealized problem configurations. The models are usually static and deterministic, and do not reflect the uncertainties inherent in real life environments. The single criterion objectives considered in such work are also far from actual manufacturing requirements in industry. Lawler et al. (1989) provide a recent review of the application of such optimization and approximation algorithms to scheduling, along with complexity considerations.

The impracticality of the above mentioned solution techniques and the need to operate in more realistic dynamic environments have led researchers to study the application of heuristic approaches. Most research along these lines has focussed on the development of various dispatching rules (Panwalkar and Iskander, 1977; Blackstone et al., 1982). These are based on job or operation related attributes and are used to sequence the set of jobs waiting to be processed on a machine. Though different dispatching rules have been noted to perform well under different criteria, they are mainly local heuristics and do not take into account system wide performance considerations. They also do not provide any direct support for rescheduling decisions (reactive scheduling).

Given their inadequacies, traditional OR based methods have not found much use in industry. Practical scheduling is largely a manual activity, effected through the experience and expertise of a handful of human schedulers. In larger and more complex environments, computerized support for decision-making is provided through tracking of raw materials and work-in-process, maintaining current shop-floor status information, and accurately generating expected requirements data. Manufacturing resource planning (MRPII, and materials requirements planning or MRPI), just-in-time (JIT) production, and optimized production timetables (OPT) are the major philosophies behind the design of such systems. These are, however, not scheduling techniques per se and only provide guidelines for controlling the production environment in general.

A constant complaint with traditional scheduling methods is that they employ an oversimplified model of the environment, resulting in schedules that need frequent changes--"ad hoc 'fire fighting' by human decision makers when unexpected situations demand change, and {resulting in} rapid deterioration of the overall coherence of organizational activity" (Smith, 1991, p. 3). Conventional techniques take an isolated view of various scheduling considerations. The lack of a means to collectively incorporate different aspects of the problem is a major deterrent to building effective and efficient scheduling systems.

2.3.2 AI Advantages

Perhaps the largest benefit in incorporating AI techniques comes through the knowledge representation facilities. Unlike the restricted OR formalisms, AI representation schemes allow a realistic modelling of the complex scheduling problem. This permits consideration of the various environmental, job and operation related attributes, constraints that need to be adhered to, preferences, and expert scheduling heuristics. Constraints play an important role in scheduling. Fox and Smith (1984b) asserts that " the crux of the scheduling problem is the determination and satisfaction of a large variety of constraints" (p. 26). The ability to take constraints into account can drastically reduce search, and this fact is used in the application of constraint-directed search methods (Smith, 1987; Bel et al., 1989). Certain constraints are relaxable under special circumstances, and heuristic constraint-relaxation techniques can also be employed. AI informed search methods, like A*, beam search, etc., provide a means for exploiting domain knowledge, and can be utilized in searching for good schedules (Shaw, 1988c; Fox and Smith, 1984). Planning has also been an active AI research area, and bears direct relevance to planning processes in schedule generation. Knowledge based monitoring software can be designed to signal the occurrence of exceptional situations. Further, knowledge based decision support frameworks can provide flexible decision aids, allowing consideration of potentially diverse preferences and goals. Developments in machine learning research hold out the possibility of adaptively and incrementally improving scheduling knowledge and strategies with experience. Automated learning methods are specially relevant for scheduling, where human analytical capabilities can often be found deficient in dealing with the complexity of the environment. Finally, AI techniques can be employed to supplement traditional scheduling methods, by making them more accessible to users or by opportunistically embedding their use in appropriate situations. A number of researchers have argued for a synergistic combination of OR and AI approaches. Simon (1987, p. 11) appeals: "instead of differentiating between OR and AI, we need to confuse, blend and synergise them as much as possible." Bel et al.(1989) conclude:

> The use of the AI approach must not lead us to reject entirely {the} traditional methods which already give good results but are insufficient to solve completely a real problem. AI enables the amount of useful knowledge for automated or semi-automated solving to be augmented, where pieces of ill-structured, but important information were neglected previously. In that sense, Artificial Intelligence does not make traditional OR methodologies obsolete, but it may bring them closer to real problems. (Bel et al., 1989, p. 244)

Kusiak (1987) provides a comparison of AI and optimization methods, and notes the potential for their integration for FMS scheduling. Effective scheduling requires the utilization of diverse knowledge sources, and search and inference mechanisms, and AI offers the possibility of their integration for the design of robust scheduling systems.

2.4 Knowledge Based Approaches to Scheduling

Representation and search are the key to AI methodologies. The first pertains to the types of knowledge included and representation scheme utilized in modelling the problem, and the second concerns how this knowledge is utilized to search for effective solutions. While expert systems and constraint based techniques have dominated research in the field, a number of AI scheduling systems use hybrid approaches seeking to combine the advantages of diverse AI and OR techniques. AI planning approaches and distributed AI frameworks have also been utilized for scheduling. Frames have, in general, been amongst the most popular of AI representation schemes and have been used in scheduling systems too (De and Lee, 1990; Blessing and Watford, 1987; Farhoodi, 1990; Sarin and Salgame, 1989; Shen and Chang, 1988; Bu-Hulaiga and Chakravarty, 1988). Traditional AI search strategies like A* search (Shaw, 1988c) and beam search (Fox and Smith, 1984) have also been employed. Many implemented systems are simulation based, and knowledge based simulation techniques have been proposed for modelling the factory environment (Shannon, 1988; Inoue and Fuyuki, 1989; Kanet and Sridharan, 1990; Manivannan and Banks, 1992). The need for the incorporation of learning mechanisms is now widely recognized, and a few learning applications, including decision-tree induction (Piramuthu et al. 1991; Nakasura and Yoshida, 1992), genetic algorithms (Davis, 1985; Hilliard et al., 1988, 1989; Syswerda, 1991), and neural networks (Zhou et al., 1990; Rabelo et al., 1990), have been reported in the literature in recent years.

In this section we summarize knowledge based approaches to production scheduling along the following categories: Rule-based systems, Constraint based search, Multi-perspective and Activity based scheduling, Other Hybrid approaches, Planning and scheduling, and Distributed scheduling. Only the basic approaches are outlined and some representative systems considered. Machine learning strategies for scheduling are discussed separately in the next chapter.

2.4.1 Rule-Based Systems

In expert or rule-based systems, the search is implicit in the representation. Knowledge is represented in the form of condition action rules, and search typically proceeds through a backward or forward chaining inference. Most scheduling expert systems use forward chaining (Smith, 1991). The objective here is to capture the expertise of human schedulers in the rule-base and emulate their decision making.

Rule based systems have been amongst the most popular of AI techniques in scheduling (Orciuch and Frost, 1984; Bensana et al., 1986; Bruno et al., 1986; Kerr and Ebsary, 1988; Lecocq and Guiot, 1988; Clancy and Mohan, 1990; etc.). Kanet and Adelsberger (1987), and Kusiak and Chen (1988) review ES applications to production scheduling. Helferich et al. (1988) recommend ES support for a number of logistic functions to improve production efficiency, and discuss ES applications to inventory management, scheduling, purchasing, customer service, transportation and packaging.

A number of researchers have, however, expressed misgivings about the applicability of pure rule based systems for scheduling. Kanet and Adelsberger (1987) question a basic premise about ES approaches:

> If an ES approach to a scheduling problem means 'mining' the knowledge from existing human expert schedulers and developing a system that the same types of schedules . . . then won't the ES produce schedules that are just as good (poor) as those produced by human experts? . . . Is such an approach not limited to providing schedules that are just as good as current manual methods with the only improvement being that they are found with less human effort . . . ? (Kanet and Adelsberger, 1987, p. 58)

Steffen (1986) similarly argues that bonafide experts just do not exist and gives nine reasons against using human schedulers as knowledge sources in developing scheduling applications. Foremost amongst his objections are that human schedulers, due to their inability in dealing with the large number of variables together, tend to follow locally greedy strategies and use artificial constraints. Blessing and Watford (1987), too, note that in an FMS setup, "there is no one to consult and glean rules from in the conventional expert system sense" (p. 83). McKay et al. (1991), however, contradict this view and write:

> The authors have supporting evidence from exploratory case study research suggesting . . . that . . . expert schedulers in a predictive sense do exist, and their ability to recognize, address, and exploit critical events is a significant component in planning and scheduling activities trying to derive feasible and desirable schedules. (McKay et al., 1991, p. 2)

They caution, however, that any human scheduler cannot be considered an expert merely on the basis of many years of experience.

A second major drawback of ES is that they often require extensive recoding in order to keep up with evolving environments, where products and processes are continually updated (Kerr and Ebsary, 1988; Steffen and Greene, 1986b). Inconsistency of knowledge across production environments is another disadvantage, considering the often prohibitive expense and effort required in ES development and validation. The incorporation of "deep" knowledge and reasoning models as envisaged for second generation ES, and addition of automated learning facilities may help alleviate these problems.

In view of these, the idea of using ES in conjunction with other AI and OR techniques has been promoted. O'Grady and Lee (1988), for example, find that ES "can be particularly rigid for control functions and there is a need for the inclusion of other AI methodologies and architectures for such functions" (p. 845). Reinschmidt et al. (1990) describe one such ES based interactive scheduler, where a linear programming (LP) optimization method handles the detailed scheduling, and ES rules guide the LP problem formulation through appropriate constraint generation and relaxation. RBD (Clancy and Mohan, 1990) and SIS (Mohan and Clancy, 1989) are examples of expert rule based dispatchers that are integrated within other factory information tracking and planning modules, to allow for real time control of production environments. Wysk et al. (1986) describe a scheme where an expert system is first employed to obtain good alternate scheduling rules, which are then evaluated through a simulation model. The performances of the scheduling rules under current system conditions are stored and later used in an inductive learning module.

LMS (Sullivan and Fordyce, 1990) is another revealing example of how rule based approaches can be effectively coordinated with other decision support facilities. It is a distributed logistics management system used for controlling manufacturing flow in IBM's Burlington, Vermont, semiconductor wafer fabrication facility. Noting a four level decision hierarchy, LMS provides for coordinated decision making amongst isolated agents. It provides a real time transaction based data gathering facility. Knowledge based data analysis and presentation provisions then make useful information available to shop floor operators. Pro-active decision support is provided through rule based heuristics which monitor system status and communicate exceptions and alerts for timely intervention. Rules are used together with a variety of current system status and job related information and global objectives, for identifying opportunities and managerial decision making. The rules used were obtained from multiple decision makers at various organizational levels. LMS has been noted to far surpass the capabilities of expert human schedulers.

2.4.2 Constraint Based Search

As mentioned earlier, the consideration of problem constraints can often restrict the search space of feasible solutions, and constraint based search techniques seek to take advantage of this. The large number of possible constraints that are relevant to scheduling can be broadly classified as necessary, preference and relaxation constraints (Noronha and Sarma 1991). Necessary constraints are those that must be satisfied to obtain a feasible schedule. The other two are not strictly necessary for feasibility, but their satisfaction can yield better schedules. Preference constraints are often used to model different scheduling objectives. Smith et al. (1986) provide a detailed analysis of the different types of constraints encountered in production scheduling.

Constraint based scheduling systems typically employ a two step approach. The first, constraint based analysis, is responsible for incrementally maintaining currently feasible solutions as scheduling decisions are being made. The second step involves the actual decision making in choosing amongst alternative schedules by searching the feasible solution space. Constraint analysis is achieved through deductive constraint propagation, that takes into account various production constraints, job and due date requirements, and any new constraints forced by the ongoing decision making. When no feasible solution alternatives exist, one may backtrack over earlier decisions or consider relaxing certain 'soft' constraints.

Constraint based schedulers differ in the degree of sophistication of the constraint analysis and propagation component, and on the nature of search used in choosing amongst feasible alternatives. Some systems employ a simple least commitment approach to constraint satisfaction, while others use specialized techniques for constraint propagation and seek to exploit specific problem characteristics. Knowledge of constraint interactions, especially interactions amongst preferences, can also be utilized to obtain better schedules. In conducting a search, many systems use a simple back-first search, with backtracking when inconsistencies are detected. Smith (1991) discusses these in greater detail. Simple search techniques have, however, been noted to be ineffective in sufficiently restricting the feasible alternatives, especially on large and complex scheduling environments. Heuristics are thus sought to be incorporated in aiding this search. Another strategy frequently employed is the hierarchical decomposition of the problem into more manageable subproblems on which to focus attention at a particular scheduling step.

A pioneering influence in knowledge based scheduling was the ISIS job-shop scheduling system (Fox and Smith, 1984). ISIS embodies a hierarchical constraint directed approach to scheduling. An order (job) based decomposition of the overall problem is used, wherein the various shop-floor orders are prioritized. Each order is thus a decision point. A schedule is then generated using constraint-directed search at the next three different levels of problem abstraction. A critical-resource capacity analysis is followed by a more detailed resource level analysis. Here, using a beam search, alternative schedules are generated and evaluated incrementally by extending partial schedules, at each step, to consider a more detailed scheduling decision. Candidate partial solutions are evaluated through their satisfaction of preference constraints (which model various scheduling objectives), and this serves as a basis for pruning the set of alternative schedules that are to be iteratively extended further. Search at this level yields a specific routing for a job, along with resource assignment time bounds. At the final level, detailed schedules are obtained through local considerations so as to minimize the work-in-process times of the jobs. Rule based constraint relaxation is utilized to redirect search when unacceptable solutions are encountered. ISIS also provides for interactive scheduling, where the system takes on an advisory role, checking user scheduling moves for any constraint violations and suggesting alternatives. ISIS was designed and run on simulated data from the Westinghouse turbine manufacturing facility at Winston-Salem but was never field tested.

Steffen and Greene (1986a, 1986b) describe another prototype system for scheduling parallel processors, using a combination of hierarchical planning and constraint-directed search. A further problem decomposition step along the time-line was undertaken in their work to reduce problem complexity. Steffen and Greene (1987) describe the application of a separate decomposition heuristic noticed in the manual scheduling of parallel machines. Noting that, in practice, specific machines are often dedicated to process certain specific jobs only for fixed time periods, they define a "dedicated-shared decomposition" heuristic to reduce scheduling effort. Arff and Hasle (1990) present a system for schedule generation using a heuristic constraint guided search described within the traditional AI state-space paradigm. Another group of constraint based scheduling systems are described in Erschler and Esquirol (1986), Erschler and Roubellat (1989) and Bel et al. (1989). Fox et al. (1989) formalize a problem solving model based on constraint satisfaction and heuristic search, and this forms the basis for activity-based scheduling in the CORTES system described in th next section.

2.4.3 Opportunistic, Multi-perspective and Activity-based Scheduling

The ISIS experiments indicated that while a job-based decomposition strategy allowed minimizing work-in-process time, it often worked against resource utilization objectives. The OPIS system (Smith et al., 1986, 1990; Smith, 1989), following from and extending the ISIS concepts, considers both job-based and resource-based decomposition. The second decomposition strategy yields subproblems that focus on the schedule of specific resources and allow resource contention amongst jobs to be resolved keeping the utilization objective in mind. Decisions on which decomposition strategy to use and what subproblem to solve at a particular point are made "opportunistically," through repeated analysis of the current solution constraints. The basic underlying premise is to first schedule bottleneck resources and thereby restrict the alternatives for the remaining decisions. OPIS takes a common view of predictive and reactive scheduling, and provides for reactive schedule management through provisions for constraint based schedule repair (Smith 1988a). The system is implemented using a blackboard style architecture that functionally distributes scheduling effort amongst different knowledge sources. Le Pape and Smith (1987) describe the techniques of constraint propagation used in OPIS. Ow (1986) compares the performance of ISIS, OPIS0 (an initial restricted version) and the COVERT dispatching rule, and OPIS is noted to outperform the other two in all the three objectives considered: minimizing tardiness costs, work-in-process time, and number of machine setups.

The order-based and resource-based problem decomposition strategies in OPIS are coarse-grained in the sense that the activities making up a subproblem are not considered as separate decision points. All activities making up a critical job or contending for a critical resource are assumed to be sufficiently and equally important to deserve immediate attention of the scheduler. The criticality of individual activities, however, depends upon both the order to which it belongs and the resource for which it competes. In incremental schedule generation, coarse grained scheduling can thus result in the consideration of less critical decisions before more important ones, and thereby prematurely constrain future options. Further, a resource is often not highly contended for over the entire scheduling horizon, and bottlenecks may shift with ongoing schedule generation. Taking a resource-based approach at a point in time and generating the entire schedule for this bottleneck can thus be undesirable.

The CORTES system takes a micro-opportunistic view to scheduling, where each activity is taken as a decision point (Sadeh and Fox, 1990; Fox and Sycara, 1990). It is based on the constrained heuristic search framework described in Fox et al. (1989). The problem space is structurally characterized as a constraint graph, and a set of texture measures defined on this structure serves to focus search on appropriate subgraphs so as to minimize backtracking. An objective function over the problem space helps in rating alternate solutions that satisfy some goal. The textures differentiate between subgraphs and measure the criticality of activities. Texture measures are related to the problem objectives using a probabilistic framework (Sadeh and Fox, 1989). Activities within an order are constrained by precedence relations defined in the process plan. Utility functions over the activity start times and resource allocations are defined based on goals, and the utilities summed over all activities provide an objective function to be maximized. Each activity is considered a variable, and the variable's value corresponds to a reservation for that activity (assignment of a start time, and allocation of resources). Scheduling proceeds through the following cycle of steps: (1) reservations are made for an activity, leading to a new state; (2) constraint propagation, as explained in Sadeh and Fox (1989), and this identifies possible inconsistencies; (3) backtracking, if inconsistencies are encountered; (4) texture measures and objectives are evaluated to identify the most crucial activity, which is to be scheduled next. Here the probabilistic model is used to define heuristics that dynamically select the activity to schedule next (variable ordering heuristics) and the commitments to be made relative to this scheduling decision (value ordering heuristics).

2.4.4 Other Hybrid Approaches

The difficulty of the scheduling problem itself, and the need to operate in an environment that is governed by a number of interrelated functions have led researchers to examine an integration of varied techniques. Some systems that use a combination of rule-based and other approaches have been mentioned in Section 2.4.1. A number of other prototype implementations have demonstrated the viability and effectiveness of hybrid knowledge based systems for shop-floor control.

Czigler and Whitaker (1990) describe a system that incorporates both knowledge based and algorithmic approaches in seeking to minimize workstation changeovers and work-in-process inventories, and meeting due-date requirements. An algorithmically generated baseline schedule is refined through a knowledge-based post processing step that attempts to resolve soft-constraint violations. The extent of rescheduling done by the post processor to obtain more desirable schedules is controlled by the speed of execution versus schedule quality tradeoffs. Reinschmidt et al. (1990) present another system where an ES guides a linear programming problem formulation, which then handles the detailed scheduling. The MULTEX system (May et al., 1991) is designed to provide intelligent decision support for scheduling through the cooperative efforts of an ES and an OR based heuristic pricing module. The pricing module is driven by a cost/benefit analysis determined heuristically from job delay costs and machine utilization prices. The schedules generated by these two sub-systems are integrated by an intelligent mediator module that preserves the crucial parts of both schedules. Lawrence and Morton (1986) provide an outline of the PATRIARCH system, which seeks to combine a heuristic scheduling algorithm with a number of AI techniques developed at Carnegie Mellon University including constraint directed reasoning, OPIS-type opportunistic scheduling, and ES.

Noting the limitations of both analytic and knowledge based techniques for effective real-time FMS control, Ammons et al. (1988) call for the incorporation of human supervisory control, with an explicit engineering of both the automated and human control functions. They emphasize the need for empirical research to identify appropriate models of human control functions, and to evaluate the effectiveness of the algorithmic and rule-based techniques when operating under human supervision.

Farhoodi (1990) provides another example of an integration of algorithmic and knowledge based techniques. A modular system architecture is described, where initial schedules are first generated using non-AI methods, and then improved upon through knowledge based evaluation and analysis. A schedule repair mechanism suggests corrective action in response to shop floor disturbances during schedule execution. The system includes a graphical user interface for effective decision support, and external interfaces to shop monitoring systems and relational databases. Frames and production rules are used for knowledge representation. Sarin and Salgame (1989) describe a similar integrated framework for knowledge based dynamic scheduling. A planning phase based on an OR approach is proposed, where a schedule is generated assuming ideal resource availability. A knowledge based operational phase then handles any schedule modifications necessitated by system status changes. Data and knowledge are represented using production rules, frames, and decision tables.

The need for an integration of AI and OR techniques is also noted in Niew et al. (1990). The authors outline a knowledge based decision support system to aid in the development of master-schedules for an IC manufacturing plant. A tentative master-schedule developed from production requirements data, is adjusted to satisfy varied objectives and constraints.

In the MADEMA system (Chryssolouris et al., 1988a, 1988b), a hierarchical structuring of the manufacturing facilities into factory, job shop, work center, and resource levels is used to decompose the control of the production environment. A multi-criterion decision making approach is implemented for work center level scheduling, where a decision is based on consideration of a number of factors with often conflicting consequences. A rule base helps determine the relevant criteria for a task, and these are then applied in the evaluation of scheduling alternatives. The system has been noted to outperform conventional dispatching rules.

The REDS system (Hadavi et al., 1990, 1992) uses a combination of least commitment planning, constraint abstraction, and a number of heuristics for real time scheduling in a VLSI manufacturing environment. A schedule is incrementally derived at different decision making levels operating over decreasing time horizons. Interfaces to real time data collection and monitoring facilities and external databases are provided, and a statistician module gives feedback about long-term problem observations like development of bottlenecks or frequent machine breakdowns.

2.4.5 Planning and Scheduling

In certain production environments, like FMS, the use of versatile machines capable of performing a number of operations vastly increases the job routing flexibility. There is, further, the need to consider supporting activities like machine setup changes and part movement across machines (often aided by an automated material handler). The decision alternatives at any point in time depend upon the current system state. The presence of computerized data collection facilities and computer control of machine operations offers a means for effective real-time monitoring of the system state and control.

The AI state space representation and planning approaches have been applied for FMS scheduling. Scheduling is viewed as a transformation from an initial state to a goal state, effected through the application of a series of state-transition operators. Systems vary in the state representation scheme and in the strategy employed for selection of the most appropriate operator given a particular state.

Shaw (1988c, 1989) employs a first-order predicate calculus representation for states, and a heuristic search procedure similar to the A* algorithm for generating schedules. A job based decomposition of the overall problem is first undertaken, and a linearly sequenced plan for each job is obtained through the heuristic search algorithm. Any conflicting resource interactions amongst the different plans are then determined, and a plan revision scheme is used to obtain modified schedules for all jobs. Different scheduling objectives can be realized through application of appropriate dispatching heuristics in guiding the A* search. Noting that no single dispatching rule heuristic will be applicable in all situations, Shaw (1989) presents an inductive learning method for opportunistically determining the best scheduling rule to apply given a particular system state. Computational results on search complexity emphasize the need for a good heuristic, and indicate that global and domain specific heuristics are better than local or general ones. Conventional dispatching rules can be embedded within the planning algorithm as the heuristic search criteria.

A similar planning algorithm for FMS scheduling is described in De (1989). Here, too, a first-order predicate calculus representation is used. Schedules are obtained in two phases, where sub-plans generated for each job in the first phase are integrated through consideration of conflicts in the second phase. Shen and Chang (1988) argue that process planning and scheduling are related issues in an FMS, and provide a frame based approach to integrate the two.

2.4.6 Distributed Scheduling

Shop floor control responsibilities are distributed across multiple decision makers, often seeking diverse objectives and operating at different levels in the organizational hierarchy. The production environment is inherently decentralized, with distinct activities undergoing concurrent execution. With the incorporation of computer controlled machinery and networking facilities, modern factories are increasingly being organized as "islands of automation". The move towards distributed manufacturing control is recent, but represents a paradigm shift from the other centralized approaches discussed above (Shaw et al., 1992).

In the design of distributed systems, scheduling proceeds through the cooperative effort of multiple "agents". The system is viewed as a network of interacting sub-systems, with individual agents controlling the local decision making. Distributed AI frameworks provide a basis for coordinating the activities of the various agents in a globally coherent manner.

Smith and Hynynen (1987) propose an architecture based on a hierarchical decomposition of the overall scheduling task. Schedules higher in the hierarchy take a more global perspective than those at lower levels that shoulder more local responsibilities, and also focus on decisions over wider time horizons and at increasing levels of process aggregation. Manager/subordinate and peer-to-peer roles are defined for schedulers in this hierarchy. The operation of each scheduler is based on the OPIS scheduler (Section 2.4.3), and a communication and coordination framework based on the propagation of locally changing constraints is proposed.

The CSS system (Ow, Smith and Howie, 1988) distributes scheduling effort in accordance with an existing organizational structure where final schedules are obtained through compromise between agents with differing objectives. A contract net framework is used, with a work-order manager (WOM) soliciting bids from resource-brokers for different operations required in a job. The work-manager is responsible for coordination of the scheduling of different operations and the completion of a job. One resource-broker is associated with each work center, and bids are based on the current schedule for this center. From the bids it receives, the WOM estimates a completion time for a job, and if acceptable, awards contracts for the bids to the brokers, who then make tentative reservations for the job. The WOM's goal is assumed to be the minimization of completion time and work-in-process time of jobs, and the brokers may have different goals (for the prototype study presented, brokers' objectives were assumed to be the minimization of production costs and queue times). Flexibility, in terms of increased decision making alternatives, is attained by allowing brokers to return multiple bids for a single call, and by having the WOM specify some slack alongwith the start and latest finish times for an operation, so as to permit the brokers some rescheduling.

Another contract net based approach for the coordination of tasks in flexible manufacturing cells is provided in Shaw and Whinston (1988). The task negotiation process amongst competing cells is modelled using augmented Petri-nets. Shaw (1988a) also discusses a bidding scheme implemented on a local area network, and compares its performance with that of certain dispatching rules. Cell level scheduling is performed by a knowledge based planning method (Shaw, 1988b).

Sycara et al. (1990) present a distributed scheduling framework based on a decentralization of the CORTES activity based scheduler (Section 2.4.3). Multiple agents make local decisions about resource assignments to activities that they are responsible for, and a complete schedule is obtained by incrementally combining the partial schedules thus obtained for an order. A pair of texture measures that quantify characteristics of the search space and help predict the impact of local decision on global objectives, and which allow agents to from expectations about requirements of other agents are defined and used to direct the distributed scheduling. A communications protocol is established for the exchange of information between agents to compute these measures and coordinate their decisions.

A market-like model for the distributed control of manufacturing flow is proposed by Lin and Solberg (1992). Real time scheduling is achieved through coordination amongst various agents, including part (job) agents, resource agents, database agents, and communication agents, which support different manufacturing activities. A job enters the system with some amount of "currency", and bargains with the resource agents for meeting its processing needs. The resource agents act as vendors and quote processing charges based on their current status. In the two way negotiation that follows, a job tries to achieve its set of weighted objectives (due dates, cost, flow time, quality, etc.), and a resource also seeks to sell its services for the maximization of its own profit and other performance criteria (utilization, etc.). A bidding strategy based on this price and objective mechanism is defined, and both parts and critical resources are given the ability to initiate bidding. The system is driven by the equilibrium of different prices and objectives. Fluctuating prices in accordance with the system load help identify and resolve shifting bottleneck problems. An object-oriented simulation system is implemented and results indicate that the price mechanism is highly adaptive to changes in system state.

CHAPTER 3
MACHINE LEARNING IN SCHEDULING

3.1 The Need for Learning

Expert systems traditionally rely on interviewing and protocol analysis techniques for the extraction of knowledge from domain experts. Human experts have, however, been found to be largely inadequate in introspectively detailing their own knoweledge. This paradox of expertise (Musen,1989), resulting in a knowledge acquisition bottleneck, is a primary deterrent to the widespread use of knowledge based systems.

In a production scheduling environment, given the human inadequacies in dealing with the vast complexity of the scheduling problem, the very existence of true experts is brought to question (see Section 2.4.1). Learning is further necessitated by the (often prohibitive) time and cost considerations in knowledge based system construction, and the need to keep up with an evolving production environment. Without any learning mechanism, ES approaches are widely perceived as unsuitable for scheduling:

> Problems like factory scheduling tend to be so complex that they are beyond the cognitive capabilities of the human scheduler. Therefore, the schedules produced by the scheduler are poor; nobody wants to emulate their performance. Even if the problem is of relatively low complexity, factory environments change often enough that any expertise built up over time becomes obsolete. (Fox, 1990, p. 81)

Yih (1990) expresses similar views regarding the lack of a source of expertise:

> In the development of expert scheduling systems, knowledge acquisition is the most time-consuming and difficult step . . . Most of the existing methods

> assume that human experts are available for the knowledge acquisition process, which is rarely the case for the scheduling problems in FMS. (Yih, 1992, p. 171)

There exists, today, a large and growing body of research on Machine Learning, directed at overcoming this knowledge acquisition bottleneck. Learning techniques seek to offer a means to acquire new knowledge, refine existing knowledge bases, and to adapt to changes in the operating environment. Though a number of researchers assert the need for incorporating learning techniques for knowledge based scheduling, efforts directed at the actual development of such systems are relatively few and recent. Machine learning paradigms considered for scheduling applications include inductive learning from examples, neural networks, case based reasoning, and genetic algorithms. Section 3.2 provides a review of learning applications to production scheduling reported in the literature. This is followed in Section 3.3 by a general discussion on the applicability of learning techniques for knowledge based scheduling.

3.2 Learning Applications to Scheduling

Thesen and Lei (1986) use a simulation based approach to construct a rule base for scheduling the operations of a single track material handling robot in an automated electroplating line. The performance of several dispatching rules under varied system conditions is obtained through simulation, and manual analysis of results indicates the best rule to use in different situations. The system state is continually monitored, and the knowledge base is used to dynamically change the decision rules depending on the current conditions. Though the expert scheduling system is noted to have surpassed the performance of any single dispatching heuristic, this approach is limited by the initially chosen set of dispatching rules and systems states considered in the simulation experiments. The authors note the need for further research to develop an automated approach for knowledge acquisition, which can yield better scheduling methods.

In view of the difficulty that human experts have in verbalizing their own decision processes, Yih (1990a) develops a method for knowledge extraction from a trace of an expert's decision sequences. This method, called Trace Driven Knowledge Acquisition, has been applied to a circuit board production line consisting of a series of chemical process tanks. A material handling robot moves jobs through the tanks, and adherence to accurate timing constraints between different chemical treatments is crucial for maintaining defect free jobs. The systems operates in three steps: (1) data collection, where a trace of decisions and corresponding system states is obtained from the simulated system; (2) data analysis, where the traces are examined to determine the scheduling rules used; and (3) rule evaluation, where the performance of these obtained rules is compared against the expert's performance. If found inadequate, the rules are fed back to the second step.

A set of variables defines the state space, which is considered divided into a number of classes. For each class, a simple decision rule determines the next course of action. The complete set of decision alternatives is provided in terms of decision rules that specify the possible actions given any state. A trace is a sequence of {existing state S_i , action taken A_j } tuples. During data analysis, the class that state S_i belongs to is obtained. In the rule evaluation stage, all decision rules in each class are scored according to the number of matches with the decision maker's actions. If the highest score is above some predefined threshold, the corresponding decision rule is assigned for that class. When the scores do not meet the minimum requirements, that class is further sub-divided, and the process repeated.

Experimented were conducted with student subjects as human schedulers, and the best amongst them taken as experts. The system-obtained rules were noted to outperform the human experts.

In a later paper (Yih 1990b; Yih and Thesen, 1991), recognizing that true experts seldom exist in real-life production environments, the authors utilize a semi-Markov modelling of the decision traces, together with an optimization procedure, to enrich the knowledge quality. Here the data analysis phase is replaced by an improvement procedure, where transition probabilities are estimated from the traces, and then an optimal decision policy algorithmically derived. These optimal policies, however, correspond only to states included in the decision making traces. Since these represent only a subset of the original state space, a generalization mechanism is necessary to extend the results and formulate rules applicable in any situation. Yih (1992) describes this third rule formation stage based on an adaptation of the TDKA data analysis method. This determines decision rules for each class of the state space, as in the rule evaluation phase mentioned above. Rule formation is, however, noted to be a bottleneck in this process, requiring a substantial human intervention. Yih et al. (1992) apply a neural network for generalization in the rule formation stage. A neural network using back-propagation learning generalizes the optimal policies so as to cover the entire problem state space.

Neural network schemes have also been applied to directly obtain near-optimal schedules. Zhou et al. (1990) demonstrate a mapping of a job shop scheduling problem into a neural net having one neuron for each operation that needs to be performed. The state of a neuron represents the starting time of the operation. The cost function employed is the sum of the starting times of the last operations of all jobs. This cost function, together with scheduling constraints, are incorporated into the energy function of a Hopfield network (Hopfield and Tank, 1985), and a linear function is derived. The network size is thus linearly scalable with increasing numbers of jobs and operations per job, and this enables the handling of large problems. Experimental results indicate that near-optimal solutions (with respect to completion times) can be obtained.

Rabelo et al. (1990) propose a hybrid architecture integrating neural network and expert system modules. The rule based modules are used to interpret high-level scheduling requests and objectives from different parts of the hierarchical FMS control structure, enforce constraints, interface with related databases, and monitor performance. They also help in developing training and retraining strategies for the neural net modules. The neural networks provide for low level learnig and production-pattern recognition tasks. An expert system module translates scheduling requirements into inputs for the neural nets, which provide performance predictions for several dispatching rules. Then, time permitting, an expert look ahead scheduler, which implements a feedback based heuristic known to generate good schedules for a tardiness criterion in an FMS, is called.

The neural networks are trained through simulation generated examples, using the back-propagation learning procedure. The system is reported to be under implementation.

Another neural network approach to learning appropriate criteria weights for multiple criteria decision making for job shop scheduling is presented in Chryssolouris et al. (1990). A three layer network using back-propagation learning is applied to learn weights to be assigned to local work-center scheduling criteria so as to meet a set of global performance objectives. A system simulation first provides the performance measure values for different operational policies (criteria weights) and varying workload levels at the work-centers, and this serves as the training inputs to the neural net. The output crireria-weights reflect the relative importance of the different criteria for attaining the specified performance levels. Three performance objectives and workloads at four work-centers considered require seven input neurons, and three output nodes each correspond to a local criteria. Ten nodes are used in the hidden layer. After training, the network is be used by specifying the anticipated workload levels and the desired performance measures, and necessary criteria weights are obtained at the output nodes.

Shaw (1989) demonstrates the use of inductive learning for adaptively changing dispatching rules in scheduling. This work is elaborated by Piramuthu et al. (1991) and Shaw et al. (1991), where they present systems that inductively learn form training examples generated through simulation. The system learns the appropriate dispatching rule to use under different factory floor conditions. A flexible production environment is characterized by a set of attributes like number of machines, buffer size, machine contention factor, etc., and simulated under varying values for these attributes. This attribute valued "pattern" together with the dispatching rule resulting in the best performance, is taken as an example to be input to the learner. The ID3 inductive learning algorithm (Quinlan, 1986) is used. A critic mechanism monitors the performance of the learned rules, and if found undesirable, moves to refine the rules by suggesting further simulation to generate more training examples. A set of meta-rules are utilized to recommend attribute values under which to run additional simulations. Experiments with the learned rules indicate improved performance than that obtained by applying any single dispatching rule.

Case Based Reasoning (CBR) is another machine learning paradigm being considered for application to scheduling problems. CBR seeks to exploit experience gained from past similar problem solving cases. Mark (1989) outlines a case based system for autoclave management. The problem here consists of loading parts of varying shapes and sizes into a large convection oven, where the process is sensitive to the part layout, and of scheduling the flow of parts through the autoclave so as to meet global requirements. Jobs arrive with different priority rankings. The manual procedure is based on the past experience of an expert operator. The proposed Clavier system seeks to obtain good autoclave loading patterns through matching the current set of jobs and priorities against a case library of previously successful layouts. The larger scheduling problem requires the identifying of salient features of past schedules, generalizing these, and a mechanism for determining which stored case best matches and is useful in the current problem context. Given the large number of interacting constraints inherent in scheduling, existing case indexing schemes are noted to be inadequate for building the case base and subsequent retrieval, and research avenues are suggested. Another case based reasoner for scheduling large scale airlift operations is proposed by Koton (1989).

Davis (1985) outlines one of the earliest Genetic Algorithm (GA) approaches for the scheduling problem. A job shop setting is considered, and a time-dependent preference list representation used. A preference list consists of a starting time at which the list goes into effect, and a sequence in which a machine prefers to process different job types. Idle and Wait operations may be included in the sequence to indicate the machine's preference to remain idle, or, if none of the specified job types are present, to wait for the next job. This representation of schedules ensures that manipulation of the lists by GA search operators yields legal schedules. Non-conventional GA operators are modelled after manual scheduling procedures that have been found to work well in the considered domain.

Hilliard and Liepins (1987) and Hilliard et al. (1988, 1989) present a series of experiments with a classifier system approach for learning job sequencing heuristics. The first two papers consider a single machine scheduling task with an objective of minimizing job lateness. The optimal strategy for this problem is to order the jobs by increasing processing time. In initial experiments, environmental detector message lists are comprised of binary coded job run-times and run-time rankings, and effectors moved jobs to specific queue locations. Different conflict resolution and credit assignment strategies are examined in Hilliard and Liepins (1987). The classifier system with this binary representation was found deficient in learning optimal rules. In Hilliard et al.

(1988), environmental detectors are modelled using high-level predicates that compare attribute values of jobs. Job processing time values, due dates, and current queue positions were considered. Each predicate represents a heuristic strategy. For example, Processing-time (job i) < Processing-time (job j) corresponds to a shorter run-time first strategy. A bubble sort routine is used to compare all job pairs, and tests on the defined predicated provide the condition part of a classifier. An action either exchanges job positions or leaves the queue unchanged. This system was able to obtain the optimal rules. The third set of experiments considered the objective of minimizing weighted tardiness, known to be an NP-complete problem. The classifier system learned rules were found to perform better than the application of any of the single heuristics alone.

Husbands et al. (1991) propose a GA based scheme where separate populations simultaneously evolve production plans for different components to be manufactured. A population member is represented as

$$op_1\ m_1\ s_1\ op_2\ m_2\ s_2\ G\ op_3\ m_3\ s_3\ op_4\ m_4\ s_4\ op_5\ m_5\ s_5\ G\ \ldots$$

where each $<op_i\ m_i\ s_i>$ set represents an operation op_i to be performed on machine m_i with setup s_i. Interdependent operations are separated into groups demarcated by G. Crossover is permitted only at the group boundaries, and mutation is also designed to ensure valid plans. In the first phase of learning, different populations evolve independently, based on the machining costs involved in the plans embodied by the population members. Different populations are ranked according to the summed costs of their members. In the second phase, the plans of equally ranked populations are simulated simultaneously. Interactions amongst the plans for different components are resolved by an arbitrator that decides which population's member to give precedence to. The arbitrator is also modelled as another population that evolves through genetic learning.

Another line of research seeks to utilize GA solution techniques for the Travelling Salesman Problem (TSP) to obtain optimal or near optimal sequences for jobs entering a production line or a specific processing facility. Special purpose genetic recombination operators for searching the space of valid tours have been developed (Goldberg and Lingle, 1985; Oliver et al., 1987; Whitley et al. 1989; Fox and McMahon, 1991). A comparison of several of these is given in Fox and McMahon (1991). Cleveland and Smith (1989) examine a number of TSP crossover operators, and compare their performance in scheduling release times of jobs in a sector scheduling problem. Though several of the operators performed well when using a tardiness objective, a more realistic work-in-process based objective is noted to indicate the limitatation of such sequencing based approaches. Liepins et al. (1987) formulate and compare certain greedy crossover operators for a deterministic single machine scheduling problem and on the TSP. Suh and Van Gucht (1987) also argue for the incorporation of problem specific heuristics into genetic search. A parallel implementation of a GA for the TSP is described in Muhlenbein et al. (1988).

Whitley et al. (1989, 1991) apply the edge recombination operator to scheduling jobs in a circuit board assembly line. Genetic search is used to obtain the loading sequence of jobs into the first machine, and a set of locally greedy heuristics then schedules the flow through the remaining machines. The performance of the final schedule thus obtained provides feedback to the GA for updating the fitness of the population members, so as to favor the generation of initial machine sequences that result in balanced optimization over the entire production line. A distributed version of the GA, using multiple populations (of smaller size), with swapping of individual members between populations after a fixed number of recombinations, is also implemented (Whitley et al., 1991) and noted to be superior to the serial GA. Comparison of the performance of schedules generated with and without the second stage application of heuristics surprisingly indicated that the GA working alone did better. The authors analyze the GA's performance on several artificially constructed problems, and conclude that "despite doing our best to torture the genetic algorithm with an extremely unfavorable problem, the results are respectable" (Whitley et al., 1991, p.369).

Syswerda (1991) address the problem of scheduling training exercises at a U.S. navy training facility, a problem bearing substantial similarity to the production scheduling environment. Here, genetic search is used in conjunction with a domain dependent schedule builder and evaluator. The GA is independent of domain knowledge and is focussed on searching the space of possible task orderings. Domain specific knowledge is brought to bear on the system through the schedule builder, which transforms the GA output sequences into legal schedules. The schedule evaluator provides feedback to the GA. This system is operationalized alongwith a schedule editor interface allowing user manipulation of generated schedules. A detailed discussion of various crossover and mutation operators is presented. The edge recombination operator is noted to perform poorly for this problem.

3.3 Discussion

Machine Learning is a relatively young area of research, and the application of learning techniques for scheduling represents one of the early moves in the field from toy problems to real world domains. The systems summarized in the previous section are thus exploratory in nature. While the need for the incorporation of learning facilities for knowledge based scheduling is recognized, the potential of different learning techniques for scheduling remains largely unexplored.

As in other application domains, inductive learning has proved useful in learning shop control policies too (Piramuthu et al. 1991; Nakasura and Yoshida, 1992). An issue of concern with inductive learning is the very large number of training examples called for by theoretical studies (Haussler, 1988; Tsai and Koehler, in press). This arises out of the need to adequately cover the example-space, and the assumption that the learner obtains examples at random (from some fixed but unknown distribution), without the aid of any domain knowledge. The use of a simulation sub-system to generate training examples may help alleviate this problem, and further research is necessary to examine ways to incorporate domain knowledge to drive the simulation so as to obtain the requisite examples for accurate learning. One such method is embodied in the critic mechanism outlined in Piramuthu et al. (1991). Learning algorithms like ID3 are also sensitive to the choice of attributes, and methods to learn appropriate features that structure the search space have been suggested (for example, Pagallo and Haussler, 1989; Nakasura and Yoshida, 1992). In complex and poorly understood search spaces like in scheduling, the identification of relevant attributes needs investigation. The use of incremental inductive learning algorithms that can work with data from actual ensuing shop floor operations is another avenue open to inquiry.

Case based reasoning offers another way of learning from prior problem solving instances. Efforts at their application for scheduling are recent and preliminary, and identifies important research questions. Mark (1989) suggests the use of inductive and explanation based schemes to generalize from past cases:

> for scheduling . . . explanation based learning techniques will be required to automatically choose and generalize salient features . . . there is a sufficient "domain model" of scheduling to allow explanation based techniques to work--not enough to learn new schedules, but enough to generalize indices that will allow the application of existing scheduling cases. (Mark, 1989, p. 179)

When the focus is on learning from the experiences of a human scheduler, behavioral characteristics of the scheduler need to be considered. In contradiction to a commonly expressed view, McKay et al. (1991) assert that human scheduling experts do exist. Based on an extensive study of manual scheduling practice, they argue that this expertise

> is based on "broken leg" cues {events or information that drastically alter the certainty of later events} or configural information found in the manufacturing environment. This human ability has the characteristics of other cognitive skills and must be accounted for, as well as taken advantage of, in computerized scheduling systems. While each cue or event may be relatively rare, there are sufficient possibilities that there are always some present and their impact should not be ignored. (McKay et al., 1991, p. 24)

The nature and role of this expertise, the process of development of expert scheduling skills, and the applicable inference strategies need to be examined and incorporated into the knowledge representation scheme employed. As part of the same study, the authors in a separate paper (McKay et al., 1989) affirm that a human scheduler's source of expertise lies in his knowledge of uncertainty. A knowledge based framework that seeks to utilize this expertise is proposed, where the human scheduler has ultimate control over generated schedules. Here, rules are used to map categorizations of the prevalent uncertainties in the shop floor to appropriate scheduling methods. The system interactively reasons about final schedule manipulations by the user, thereby learning to refine its own knowledge. Though no definite learning procedures are mentioned, the application of machine learning techniques holds promise.

Research on learning approaches to scheduling also needs to focus on ways to incorporate and effectively tradeoff the various, often conflicting, management objectives, and facilitate the inherently decentralized decision making involved in the production environment. The validation of learned knowledge is another important issue to be addressed. The mostly simplistic knowledge representation schemes used in machine learning systems also need to be enhanced for practical scheduling applications. Further, the dynamic nature of the scheduling environment calls for real-time learning methods, and systems relying on extensive experimentation may prove inappropriate. Further research is necessary to understand and investigate the limitations of currently employed learning strategies, and realize the full potential of machine learning techniques.

CHAPTER 4
GENETIC ALGORITHMS

4.1 Overview

Genetic Algorithms provide a stochastic search technique inspired by principles of natural genetics and evolution. They operate through a simulated evolution process on a population of string structures, each of which represents a candidate solution in the search space. Evolution of populations involves two basic steps: (1) a selection mechanism that implements a survival of the fittest strategy, and (2) genetic recombination of the selected high fitness strings to produce offspring for the new generation. Selection ensures that above average strings contribute to a greater number of offspring in the next generation (on average).

GAs are considered suitable for application to complex search spaces and combinatorial optimization problems, where a balance is often sought between full exploitation of the currently known solutions and a robust exploration of the entire search space. GAs provide an effective means for managing this tradeoff. It has been noted that GAs derive their power

> from a very simple heuristic assumption: that the best solutions will be found in regions of the search space that contain relatively high proportions of good solutions; and that these regions can be identified by judicious and robust sampling. (Booker, 1987, p. 61)

The selection scheme operationalizes exploitation, and the recombination operators effect the exploration of the search space. Goldberg (1989a) points out fundamental differences between GAs and other search procedures:

> 1. GAs work with a coding of the parameter set, not the parameters themselves.
> 2. GAs search from a population of points, not a single point.
> 3. GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.
> 4. GAs use probabilistic transition rules, not deterministic rules. (Goldberg, 1989, p.7)

Section 4.2 provides an overview of genetic search. Here, some basic genetic search issues are examined, including the string representation, the reproduction and recombination

operators, the schema theorem, and certain factors that can at times make a GA diverge from the global optimal. Machine learning with genetic algorithms and classifier systems are introduced in Section 4.3. Recent work by Vose and Liepins (1991) provides a detailed characterization of GA search behavior, and in Section 4.4 this theoretical framework is examined.

4.2 Genetic Search Issues

4.2.1 Representation

As mentioned above, GAs use an encoding of the search space attributes. The coding scheme is critical to the success of GAs, and a binary string representation has been traditionally used. The use of a binary representation is advocated by the principle of minimal alphabets (Goldberg, 1989a), which maintains that lower cardinality alphabets foster the parallelism that is implicit in genetic processing. As a consequence, most theoretical studies have also been conducted in terms of a binary scheme.

One approach to considering higher level representations in genetic search is to interpret binary substrings as primitives of the high-level language (Forrest, 1985). The direct utilization of a nonbinary representation scheme, however, provides greater intuitive appeal for practical applications, and a number of researchers have reported success with complex string representations (Davis and Coombs, 1987; Grefenstette, 1989). Modified operators for high-level representations have been proposed (Antonisse and Keller, 1987; Grefenstette, 1989). Some theoretical work with nonbinary string encodings has also been reported (Antonisse, 1989; Goldberg, 1990b; Wright, 1991).

4.2.2 Selection, Crossover and Mutation

The evolution of populations from one generation to the next is effected through three basic biologically inspired operators: selection (reproduction), crossover, and mutation.

Selection is the GA analog of asexual reproduction, and embodies a survival of the fittest strategy. It is implemented by evaluating each population member against the search space objective function and using these fitness values as a basis for determining the number of offspring that each member will contribute to the new population. The selection algorithm converts the expected number of offspring contributed by an individual, as manifest in its fitness value (a real number), to an integer of actual number of offspring that are obtained. As such, it incorporates some selection bias, and can lead to premature convergence problems.

A variety of selection strategies have been suggested in the literature, and Baker (1987) and Schaffer (1987) analyze the selection bias of the different schemes. Proportionate reproduction (Goldberg, 1989a) is the most common, and chooses population members for creating offspring probabilistically, based on their fitness values. Goldberg and Deb (1991) provide a analytic comparison of four popular selection methods: proportionate reproduction, ranking selection (Baker, 1985), tournament selection (Goldberg, Korb and Deb, 1989) and selection in steady-state GAs as employed in the Genitor program (Whitley, 1989).

Crossover is the GA analog of sexual reproduction, and implements a mating scheme between two "parent" strings to produce progeny that carry the characteristics of both parents. The crossover operator is invoked, with some fixed probability, after the selection procedure chooses two reproducing population members. In the simple single-point operator, a crossover site is selected uniformly at random, and parent substrings on either side of this site are exchanged to form two offspring for the next generation. Multi-point crossover (Spears and DeJong, 1991) creates more than two substrings for exchange in each parent, and is thus more disruptive. In uniform crossover (Syswerda, 1989), each string position is randomly considered for exchange with a probability of .5. Eshelman et al. (1989) compare different crossover operators, and DeJong and Spears (1990) determine that choice of an appropriate operator is also related to the population size.

Mutation is applied with low probability to perturb the values of each string entering a new population. It is a secondary operator, and its primary purpose is to insure against a population converging to uniformity and thus becoming incapable of further search. It also ensures that no string values are permanently lost from a population.

4.2.3 The Schema Theorem and Implicit Parallelism

Central to an understanding of GAs is the notion of schemata. A schema is a set of strings having certain fixed values at specified positions in the strings. For example, considering a binary encoding, the schema H: *1**0* (here, a * represents a wild card, in that we can have either a 0 or 1 in its place) represents all strings that have either a 0 or 1 where a * occurs. A schema represents a region of the search space, and individual strings are instances of a schema. The order of a schema, denoted o(H), is the number of defined bit positions. A schema's defining length, designated $\delta(H)$, is the distance between the outermost defined bit positions. In the above example, $\delta(H) = (5-2) = 3$, since bits positions 2 and 5 (from the left end) contain defined values.

The effect of crossover is to disrupt schemata. When a crossover position falls between a schema's defining length positions, unless the two parents are instances of the same schema, it produces offspring that represent different schema. Mutation also has a disruptive effect on schemata. This disruption caused by different operators is often taken as a basis for their comparison (Spears and DeJong, 1991; Eshelman et al. 1989).

Let m(H,t) be the number of instances of a schema H in a population at time t, and let f(H) be the average fitness of all strings that are instances of H. If $\bar{f}$ represents the average fitness of the entire population, then the effect of reproduction on schema can be expressed as

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}}$$

Thus, a schema with higher than average fitness will have an increasing number of samples (instances) in the next generation. Overall, reproduction leads to a reduction in the number of distinct schemata present in successive populations.

Crossover and mutation, on the other hand, seek to produce new schemata through their disruptive effects. After accounting for crossover and mutation, invoked with probabilities of p_c and p_m respectively, the following relationship can be derived (Goldberg, 1989):

$$E[m(H,t+1)] \geq m(H,t)\frac{f(H)}{\bar{f}}.[1-p_c\frac{\delta(H)}{L-1}-o(H)p_m]$$

where l is the fixed string length and E denotes expectation.

This inequality represents the famous Schema Theorem, also called the Fundamental Theorem of Genetic Algorithms. It provides a lower bound on the expected number of instances of a particular schema in the next generation, under proportional reproduction, single point crossover, and uniformly random mutation. It indicates that highly fit schemata with low order and short defining lengths receive an exponentially increasing number of its instances in successive populations. Such low order, short defining length schemata are considered "building blocks" that combine to form high performance strings. Bridges and Goldberg (1987) have extended this analysis and provided an exact expression for the expected proportion of a particular string in the next generation.

It has been estimated (Goldberg, 1989a) that the effective number of schemata that are processed in a single generation is of $O(n^3)$, where n is the population size. This demonstrates the implicit parallelism that is noted to be an inherent part of genetic search, and a primary source of its power and effectiveness. GAs can thus exploit information about the utility of a very large quantity of schemata while working with a relatively small number of strings, and that too, without any burden of extra memory or book-keeping functions.

4.2.4 GA Failure Modes

Though providing a robust search mechanism, GAs are prone to failure under certain conditions. A number of factors have been attributed to this inability of GAs to converge to global optima (Liepins and Vose, 1990, for example). Amongst these, deceptiveness, sampling error, schemata disruption due to crossover, and inappropriate choice of encoding scheme have received considerable attention.

The notion of deceptiveness arises from the Schema Theorem's prediction that low order schemata with high fitness will have increasing proportions of their instances in subsequent generations. Higher order schemata, representing improving solutions, are formed out of these low order building blocks. Deceptiveness occurs when the best low order schemata do not comprise the best high order schemata. The search trajectory is then led astray. Deceptiveness results from the manner in which a particular coding scheme maps to the function values, and such functions are considered 'hard' for GAs.

Goldberg (1987) analyzes a 2-bit deceptive problem (the minimal deceptive problem) and finds that in spite of deliberate efforts to mislead it, the GA usually converges to the global optimal, unless severely impeded by the initial population bias. Fully deceptive function of higher order (> 2 bits) do, however, exist and have been constructed using Walsh transforms (Goldberg, 1989b, 1990c). A rigorous treatment of deception is presented in Goldberg (1989), which also provides an algorithm for determining the deceptiveness of a function. Liepins and Vose (1990) show that full deceptiveness exist for all representations of length greater than two, and that these can be transformed to fully easy functions through a representational chance, thus emphasizing the importance of selecting appropriate GA encoding schemes for problems. They also report that linear invertible transforms may be used for this purpose. Goldberg et al. (1989, 1990) develop a nontraditional genetic search procedure, called messy GAs, and report preliminary studies indicating success at overcoming deceptiveness.

Sampling error occurs when a GA is unable to accurately estimate the relative utilities of competing schemata. As mentioned earlier, due to its implicit parallelism, the number of schemata processed is much higher than the actual number of samples (population members) used. A schema's utility is estimated by the average fitness of its instances present in the population, and the use of small population sizes can give rise to errors in the estimates of relative schema utilities. When the fitnesses of the current population members do not accurately reflect the true average schemata fitness, certain useful schemata may not be carried over into the following generations. The cumulative effect of such sampling errors often lead a GA to prematurely converge to sub-optimal solutions. Eshelman (1991) studies the implications of the disruption of different crossover operators and certain selection schemes on sampling error. Other means to ameliorate this problem (Liepins and Vose, 1990) are through increasing population sizes, thoroughly randomizing initial populations, and maintaining a single copy of each individual string as done in the GENITOR algorithm (Whitley, 1989).

The disruption of schemata by crossover may pose problems for a GA even when the function itself is not deceptive. This happens when the form of crossover applied is not conducive to obtaining good solutions from smaller building blocks. The Schema Theorem indicates the greater susceptibility of longer defining length schemata to crossover disruption. The inversion operator has been suggested for altering a string's defining length, but has had mixed results (Goldberg, 1989a). Arguing that the classical definition of schema may not be valid under alternative crossover operators, Vose and Liepins (1990) provide a generalization of the schema theorem, which helps characterize building blocks in terms of the particular form of crossover being used.

In the absence of prior information about the search space, little guidance is available on the choice of an encoding scheme. Schaefer (1987) describes a system that adaptively modifies the string representation. Schrandolph and Belew (1991) also propose a dynamic scheme for encoding parameters of a real-valued search space to binary strings.

4.3 Machine Learning and Classifier Systems

A learning system can be characterized as one that is capable of making structural changes to itself over time. Genetic Algorithms may be used for searching the space of legal structures so as to achieve some desired performance level or effect certain behavioral changes. GA based machine learning allows for both knowledge acquisition and refinement, and supports a production system architecture enabling parallel rule-activations. The basic GA-based learning system comprises of a task subsystem, whose performance is to be improved over time, and a learning subsystem (DeJong, 1988). The performance task is generally controlled by a rule based production system, and genetic search operators manipulate rule structures in seeking to learn useful knowledge required by the task subsystem.

GA based machine learning systems usually follow either of two basic approaches. In the Pitt approach, each individual in a population represents a body of rules (that is, an entire production system) that together execute the performance task. Alternatively, the Michigan approach considers each population member as a single rule, and it is these individual rules, rather than entire rule-sets, that are manipulated by the genetic operators. Systems operating on the latter approach are generally referred to as Classifier Systems.

Rules (or classifiers) are message based and operate on a system-wide message list that serves as a working memory. External conditions are reflected on this list through environmental detector messages. The condition part of each rule consists of a pattern that is checked for a match with any of the entries in the message list. The action part of a rule specifies a message to be posted to the list and a rule "fires" when its condition part matches some posted message. The system interacts with the environment through a set of effectors that are activated by posted messages. Any changes in the environment can thus result in a sequence of rule firings in response to message list updates, and may bring about external actions implemented through the effectors.

Learning is driven by a feedback mechanism that provides rewards or penalties to the classifiers based on the desirability of any actions taken. Since each rule-firing need not define an external action, and since all actions may not result in payoff, an arbitrary number of rules may fire between two payoff events. The proper distribution of payoffs amongst the rules thus becomes a critical issue. Each rule carries a fitness value reflecting its performance effectiveness, and a conflict resolution policy resolves firing contentions between the rules. Conflict resolution implements a competition amongst rules for control of the system's actions, and is based on a bidding scheme where each classifier bids an amount related to its fitness. The operation of a typical classifier system thus follows a message matching-bidding-payment cycle. Genetic learning is invoked after some predetermined number of such cycles.

The popular credit assignment scheme for classifier systems is the Bucket-Brigade Algorithm (Holland, 1985). Here, all rules that are active at payoff time have their strengths incremented. Fitness is then also re-distributed from these rules to others that had posted messages which, in turn, had enabled the firing of the rules. This process continues until credit is distributed across the entire chain of classifiers whose firing resulted in the payoff action. Over long sequences of classifiers, however, this procedure is rather slow, and requires several strength reinforcement cycles before early rules in a sequence receive adequate credit. Riolo (1987a, 1987b) discusses a number of issues aimed at making this procedure more effective. Liepins et al. (1991) examine several other credit assignment achemes. Grefenstette (1988) suggests a combination of the Michigan and Pitt strategies, where the bucket-brigade is used to assign payoff to both individual rules as well as rule-sets. A hierarchical credit assignment scheme is described in Wilson (1987). Another enhanced bucket-brigade mechanism is presented in Huang (1989).

A stochastic bidding scheme is generally employed in classifier systems and bids are based on a rule's fitness value. Goldberg (1990a) suggests a variance sensitive bidding procedure that takes into consideration the variance of a rule's received payoffs. Rules with more reliable payoff are treated more deterministically, while those whose payoffs are uncertain (greater variance) have a larger stochastic component. Robertson and Riolo (1988) note that classifier systems tend to produce a large number of over-generalized rules, and thus make bids a function of a rule's specificity too.

Based on empirical studies, Robertson and Riolo (1988) further report that, contrary to existing theory, an increase in population size results in improved learning. They also develop certain mechanisms to promote the formation of good rule sequences. Wilson and Goldberg (1989) examine a number of problems with classifier systems. They argue for the adoption of a modular rule organization that eliminates the need for long rule-chains. A merger of the Pitt and Michigan approaches, akin to that used by Grefenstette (1988), is suggested.

An attractive feature of classifier system learning is the development of default hierarchies of knowledge. Holland (1992) elaborates:

> Shortly after the system starts running, it evolves rules with simple conditions--treating a broad range of situations as if they were identical. The system exploits such rules as defaults that specify something to be done ion the absence of more detailed information. Becausethe default rules make only coarse discriminations, however, they are often wrong and so do not grow in strength. As the system gains experience, reproduction and crossover lead to

> the development of more complex, specific rules that rapidly become strong enough to dictate behavior in particular cases.
>
> Eventually the system develops a hierarchy: layers of exception rules at the lower levels handle most cases, but the default rules at the top level of the hierarchy come into play when none of the detailed rules has enough information to satisfy its conditions. Such default hierarchies bring relevant experience to bear on novel situations while preventing the system from becoming bogged down in overly detailed options. (Holland, 1992, p.71)

The learning of such default hierarchies is desirable since they are more parsimonious than non-overlapping rule sets, promote more efficient learning, and also provide a more intuitive and natural organization of knowledge (Goldberg, 1989a).

In adapting to changing situations, a classifier system can lose certain inactive rules which may have been accurately learned and may perform well in the future. In order to retain such currently inactive, but useful rules, Zhou (1990) develops a long-term memory for cumulative learning. Knowledge in this long-term memory is managed in related chunks of rules, and organized in a hierarchy. A generalization mechanism is employed, and the author also addresses issues like indexed storing of rules, the context based retrieval of these rules in the future, and strategies for the replacement of rules in the long-term memory.

Though a standard classifier system operates with binary strings, a number of more expressive representations have been successfully used. Booker (1991), for example, describes schemes to map high-level attributes to binary substrings. Using similar concepts, Forrest (1985) translates semantic network constructs into a classifier system format. Grefenstette (1989) directly uses a high level representation, and points out several advantages over binary encodings:

First, it makes it easier to incorporate existing knowledge, whether acquired from experts or by symbolic inductive learning programs. Second, it is easier to explain the knowledge learned through experience. Third, it may be possible to combine several forms of learning in a single systems. (Grefenstette, 1989, p. 343)

4.4 A Detailed Characterization of GA Search Behavior (The Vose-Liepins model)

A rigorous mathematical formalism for a simple genetic algorithm has been provided by Vose and Liepins (1991). Modelling recombination through crossover and mutation as a dispersion operator, and selection as a focussing operator, they obtain a precise characterization of the punctuated equilibria phenomenon often noticed in genetic search--alternating periods of rapid evolution and generations of relatively stable populations. GAs are considered as dynamical systems in a high dimensional Euclidian space, and expected population trajectories are obtained assuming infinite populations. As in many other theoretical studies on GAs, Walsh matrices provide a basis for their analysis too.

Binary strings of a fixed length L are considered, and $N=2^L$ represents the total number of possible strings. The set of binary strings is also identified with integers from 0 to (N-1) . A vector $s^t \in \mathbb{R}^N$ models the t^th generation, with the i^th component s^t_i being the probability of i being chosen for reproduction. Another vector $p^t \in \mathbb{R}^N$ has i^th component equal to the proportion of i in the t^th generation. The probability of two strings i and j combining to produce a new string k is represented by $r_{i,j}(k)$. Then, the expected proportion of a string k in the next generation is

$$E[p_k^{t+1}] = \sum_{i,j} s_i^t s_j^t r_{i,j}(k) \ .$$

Assuming infinite populations, the law of large numbers gives

$$p_k^{t+1} \rightarrow E[p_k^{t+1}] \ .$$

For binary recombination through crossover and mutation, the following relationship is noted to hold: $r_{i,j}(k \oplus l) = r_{i \oplus k,\, j \oplus k}(l)$, which gives $r_{i,j}(k) = r_{i \oplus k,\, j \oplus k}(0)$, where $\oplus$ denotes the exclusive or operator. Given $r_{i,j}(0)$ for all (i,j) combinations, all the recombination probabilities $r_{i,j}(k)$ are thus obtainable. The matrix M is defined with its (i,j)th entry $m_{i,j} = r_{i,j}(0)$. This probability of any two strings i and j recombining, through single point crossover and uniformly random mutation, to give the string 0 is found to be

$$M_{i,j} = (1-\mu)^L[\eta^{|i|}(1-\chi+\frac{\chi}{L-1}\sum_{k=1}^{L-1}\eta^{-\Delta_{i,j,k}}) + \eta^{|j|}(1-\chi+\frac{\chi}{L-1}\sum_{k=1}^{L-1}\eta^{\Delta_{i,j,k}})]$$

where μ and χ are the mutation and crossover probabilities respectively, and $\eta = \frac{\mu}{1-\mu}$.

Here, $|i|$ is the number of 1s in the bit vector corresponding to the integer i, and

$\Delta_{i,j,k} = |(2^k-1) \otimes i| - |(2^k-1) \otimes j|$.

An operator $\tilde{M}$ is defined as

$$M(s) = < (\sigma_0 s)^T M (\sigma_0 s), \ldots, (\sigma_{N-1} s)^T M \sigma_{N-1} s) >^T$$

where the $(\sigma_j\ s)$ denote the permutation.

$$\sigma_j < s_0, \ldots, s_{N-1} >^T = < s_{j \oplus 0}, \ldots, s_{j \oplus (N-1)} >^T$$

A second operator F is a non-negative diagonal matrix with $F_{i,i} = f(i)$, the objective function value for the string corresponding to the integer i. Then, Fp^t is a vector that points in the same direction as s^t, that is,

$$s^t = \frac{1}{\lambda} F p^t, \qquad \text{where} \qquad \lambda = \frac{1}{\sum_i f(i)p(i)} \quad .$$

Using ~ to denote the equivalence relation defined by $x \sim y$ if and only if $\exists\, \lambda > 0$ such that $x = \lambda y$, we have, $s^t \sim F p^t$.

Further, $\bar{M}(s)$ is a vector that has its ith component equal to the expected proportion of i in the next generation, that is, $E[p^{t+1}] = \bar{M}(s^t)$. Assuming infinite populations, we may write

$$p^{t+1} = \bar{M}(s^t).$$

These operators form the basis of the Vose-Liepins model, and helps characterize GA search through he matrices F and M. An exact representation of the limiting behaviour of a simple GA, as the population size tends to infinity, is given by the relation

$$s^{t+1} \sim F\, \bar{M}(s^t)$$

Considering G to be the composition of the operators F and $\bar{M}$, the progression of GA search from one generation to the next is given by the iterates of G, and convergence corresponds to the fixed points of G.

It is shown that the only stable fixed point of F is that associated with the maximum value of the objective function. The fixed points of the quadratic operator $\bar{M}$ have been studied through a matrix M*, with (i,j)th entry $m_{i \oplus j, i}$, related to th differential of $\bar{M}$, and it is proved that if M* is positive and has its second largest eigenvalue less than 1/2, then every fixed point of $\bar{M}$ is stable. An explicit expression of the spectrum of M* has been derived by Koehler (1992), and the second largest eigenvalue is noted to be $(1/2 - \mu)$. All fixed points of $\bar{M}$ are thus asymptotically stable when the mutation rate is between 0 and 0.5. Further, a conjecture relating to dynamical systems is noted to imply that the fixed point is unique, and corresponds to a population with equal proportions of all members. Recombination is thus viewed as a dispersion or diffusion-like operator.

The evolution of populations through the operators F and $\bar{M}$ help explain the punctuated equilibria observed in genetic search. Populations move under the influence of F towards one of its fixed points. If this is not maximally fit, then it is an unstable fixed point, and recombination will ultimately cause a major changre in the population and move it towards another fixed point of F.

Nix and Vose (1992) derive an exact model of a finite population GA as a Markov chain, and show that for large populations, the trajectory of this model follows very closely, and with high probability, that of the infinite population model. With nonzero mutation, a finite-population corresponds to an ergodic Markov chain, and thus has some finite probability of visiting every state. The steady state distribution is, however, shown to concentate probabilities near the fixed points of the infinite population model (which correspond to local optima). This implies that the GA will move from one local optimal to another in the process of its search. Vose (1993) extends this analysis through a goemetric interpretation of genetic search trajectories in both the infinite and finite population cases.

CHAPTER 5
AN ADAPTIVE, DISTRIBUTED DECISION SUPPORT SYSTEM FOR PRODUCTION SCHEDULING

5.1 Overview

Information systems designed to aid in decision making in semi and unstructured problem environments typically consist of three basic components: a database management system providing for data modelling, management, and manipulation, a model management system that includes various quantitative models and provides analytical capabilities, and a control and dialog subsystem through which the user can communicate with the system.

In recent years, Expert System approaches have been incorporated in such decision support systems (DSS). Knowledge based functionalities greatly enhance DSS capabilities, and seek to aid the user in the appropriate selection of models, and in their integration and application under varied problem situations. Knowledge acquisition has, however, proved to be a major impediment, especially in the relatively unstructured domains that such systems are sought to be applied to. There is, further, the need to adapt the decision support knowledge and capabilities to evolving external environments. This calls for continuous and incremental knowledge refinement methods. Without an efficient and effective learning mechanism, knowledge based decision support is largely infeasible in complex and dynamic environments.

Here, an intelligent decision support framework is described, that includes an automated learning component supporting both knowledge acquisition and refinement. Genetic Algorithm based learning is used, and the system is developed for application to a production scheduling environment. More specifically, it is designed to aid in the scheduling of parts in a manufacturing facility.

The manufacturing environment is inherently distributed (Sycara et al. 1990; also see Chapter 2). Modern factory floors are divided into distinct work-areas, and are controlled by agents with largely independent decision making responsibilities. The completion of a manufactured part requires performing several operations across various work-areas. Furthermore, the routing of different parts across the work-areas may differ too. Schedules for parts to be manufactured are developed incrementally, and the flow of parts along the different work-areas is maintained through the coordinated efforts of the various schedulers. Dispatching decisions, at the work-areas, are made in consideration of locally relevant criteria, but should also be consistent with system-wide objectives. The system should thus foster the integration of, and cooperation amongst the decision making activities in this de-centralized environment, in a globally coherent manner.

Scheduling goals may vary widely across different dispatchers, and global performance objectives can also specify multiple, often conflicting requirements (see Section 2.2). The DSS design should thus be able to incorporate a wide variety of performance criteria, and facilitate coordinated decision making by obtaining a balance amongst these objectives.

The system described here employs an Object-Oriented modelling of the environment. The various factory floor entities, including work-areas, queues, servers, and jobs (parts to be manufactured) are implemented as object classes. Performance criteria, the dispatchers, knowledge bases, knowledge representation primitives, and the genetic learning operators, are also modelled using objects and associated methods.

The distributed decision support framework controls the job shop using multiple intelligent objects as elaborated in Section 5.1. A simulation sub-system models the decision making environment. Such an embedded simulation sub-system provides for the incorporation of a source of deep reasoning within the DSS, as sought in second generation expert systems. It also allows for the use of the system under different environmental setups.

The knowledge representation scheme developed allows for the consideration of various factory floor attributes. Standard dispatching rules from the sequencing and scheduling literature and other algorithmic models that designate jobs for dispatch, as well as any heuristics that human schedulers may have found useful, modelled in the form of conventional condition-action rules, can also be considered in this representation. Each rule in this scheme represents a job-dispatching strategy that can be used under specific environmental conditions. The learning sub-system operates on this same knowledge representation, which thus also has to be amenable to manipulation by the different genetic search operators. Specialized operators are designed for genetic learning within this high-level representation.

The basic decision support architecture is described in Section 5.2 and Section 5.3 gives an overview of the system organization in a scheduling environment. In Section 5.4, the knowledge representation scheme is presented. The inference scheme is detailed in Section 5.4. Genetic learning details are considered in Section 5.4. Here, the recombination operators, the credit assignment scheme are explained.

5.2 System Architecture

The basic system organization is shown in Figure 5.1. A simulation model embodies the distributed decision making environment. An Object-Oriented knowledge base integrates the data, models, and production system rules relevant to the performance tasks. The rules use a high-level representation suitable for genetic learning, and each rule implements a decision-making strategy. The inference module effects conflict resolution among the various rules competing for control of the simulated performance tasks. It also provides credit assignment for the genetic algorithm based learning component.

As indicated earlier (Section 4.3), a genetic algorithm based machine learning system is comprised of two basic sub-systems: a task environment whose performance is to be improved over time and a learning module. Here, the simulation model serves as the performance environment. The Genetic Algorithm maintains a population of competing strategies, that evolves with each learning episode. It is these population rules that operate the decision making in the simulation. Learning is initiated at pre-determined intervals of time. A long-term memory prevents the loss of rules that may not be competitive

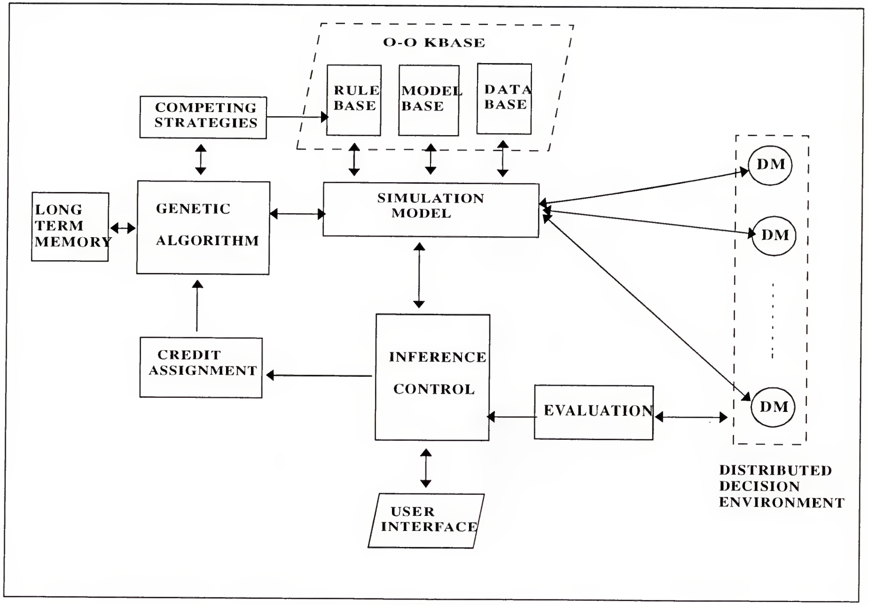

Figure 5.1: System Architecture

under current conditions, but which have performed well in the past and so may prove useful in the future.

Decision making is envisioned as a distributed activity that may involve multiple steps, and a number of decision makers. At the completion of a decision activity sequence, the evaluation module provides a numeric payoff estimate that reflects how well global objectives have been met. Intermediate payoff points, indicating the satisfaction of local decision objectives, may also be established. These payoff values are used by the inference mechanism to distribute credit amongst the various rules that were invoked for the decision making.

The distributed nature of the system is illustrated in Figure 5.2. Each decision maker may be viewed as possessing its own knowledge base, and genetic learning is also conducted independently for the various decision making entities. Decisions are evaluated in accordance with predefined managerial objectives. Such evaluations can be performed after the completion of a task, or may be interspersed at different points within the decision sequences required of the task. The distribution of payoff amongst the different learning modules is determined by the credit assignment scheme.

Considering an Object-Oriented design, decision support responsibility in this framework can be viewed as being distributed across multiple intelligent objects. An object is considered intelligent if it carries a knowledge base, and takes actions based on inference from the rules contained therein. Both the decision making and evaluation entities of Figure 5.2 can, in general, be considered as intelligent objects. Evaluation objects can be viewed as management entities embodying decision-making objectives, and may possess their own knowledge bases with the rules embodying tradeoffs between possibly conflicting goals. This knowledge determines the payoffs associated with the decision-making alternatives. The learning of knowledge related to these tradeoffs in different environments is an important issue too, though not addressed in this research.

Genetic learning is critically dependent on credit assignment, and the formulation of payoff functions and the way it is distributed across decision-makers will largely determine success in learning. The developed framework seeks to incorporate the flexibility of including multiple evaluation objects at different points in the decision sequence. Accurate learning under such a distributed credit assignment scheme will, however, require coordination amongst different payoff sources so as to enable the effective modelling of the relative importance of different evaluation criteria to a particular decision maker.

Adequate credit apportionment can, in general, involve another knowledge based component in the system. In complex environments, this may call for a learning mechanism to determine an appropriate payoff distribution policy, consisting of a weighted average of different evaluation values for each genetic learning component. Such a second level of learning, however, is not implemented in this research. and a pre-determined credit assignment policy is assumed.

The architecture is implemented for a production scheduling environment. This is described in the next section.

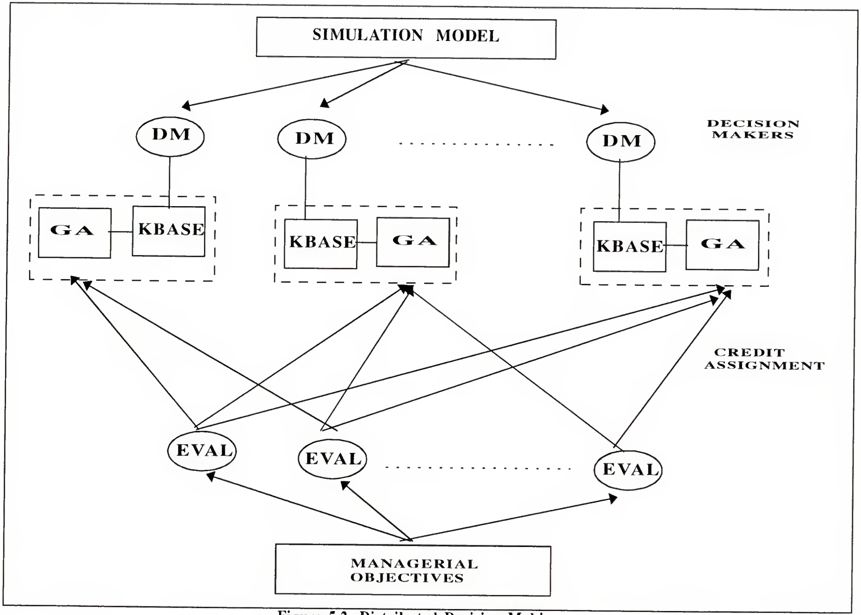

Figure 5.2: Distributed Decision-Making

5.3 Production Scheduling Environment

A conventional job-shop scheduling problem is considered. An example of a typical factory-floor setup, adapted from Chiang et al. (1990) is shown in Figure 5.3.

Production resources (servers, or machines) are located in work-areas (machine banks), each of which consists of a number of resources. Similar resources are grouped into a work-area such that only a single type of operation can be performed in a particular work-area. Servers within a work-area can, however, possess different characteristics (varying processing times, operator skill levels, quality, etc). Each such work-area is fed by a single queue that holds the jobs waiting to be processed. Individual resources may serve only a single job at a time, and no operation on a job can begin before the preceding operation is completed.

Jobs entering the system are one of several different job-types, and the completion of each job requires the performance of several operations on the work areas. The different job types have their own set of alternative routings through the shop, and a job is assumed to enter the system with a predefined process plan. For example, two different jobs-types may follow the routes:

Job-type_1: WA_1, {WA_4 or WA_5}, WA_7, WA_9, WA_10, WA_11.

Job-type_2: WA_1, {WA_5 or WA_6}, WA_11.

Routing flexibility may exist even for the same job-type. Here, after completion of the first operation in WA_1, a job belonging to Job-type_1 can undergo processing in either WA_4 or WA_5. The processing times for different job-types vary across servers within a work-area. Whenever a server is assigned to process a job that is not of the same as

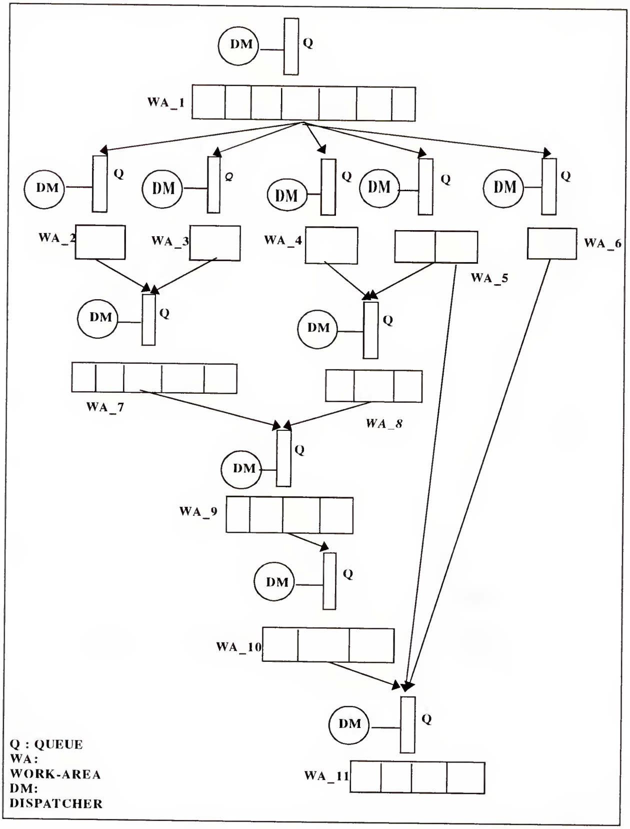

Figure 5.3: Factory-Floor Setup

that of the job it has most recently processed, it requires a certain setup-time to prepare for the different operation. Setup intervals are also needed when a server first begins processing. A server may not process any job for the duration of the setup.

The decision makers in this environment are the dispatchers responsible for selecting jobs for service from amongst those currently in the queues. Each queue has a dispatcher associated with it , and dispatchers carry their own knowledge-bases.

To provide for routing flexibility in a job and yet ensure that it follows the best path through the work-areas under existing shop-floor conditions, the system implements a "virtual job" mechanism. Whenever alternative processing routes are available, and a decision needs to be taken regarding the assignment of a job to one of several work-area queues, dummy copies of the job are placed in all the possible alternative queues that the job may enter. Once selected for dispatch by any of the queue-dispatchers, the virtual jobs are removed from all these queues.

Evaluation objects may be established at different points in the operation sequence. The credit assignment scheme will determine the payoffs that different dispatchers receive from the individual evaluation objects.

5.4 Knowledge Representation and Inference

5.4.1 Representation

The effective modelling of the shop floor environment calls for a rich knowledge representation scheme. Bel et al. (1989) distinguish between static and dynamic knowledge requisites for scheduling. Static knowledge refers to information about various factory floor entities, as well as processing times and sequences, due dates, and performance objectives. Dynamic knowledge denotes the expertise used to obtain feasible and good schedules, and includes (1) theoretical expertise from OR research, (2) empirical expertise obtained through using dispatching rules in simulations, and (3) practical dedicated expertise provided by human schedulers and shop floor managers. Given the widespread opinion that true scheduling experts just do not exist in many production environments (cf. Section 2.4.1), it is imperative that the representation used in a machine learning system facilitate the learning of rules from knowledge primitives. These primitives should also seek to include as many of the dynamic knowledge constituents as possible.

The knowledge representation scheme has been developed with a view to incorporating the different aspects of scheduling knowledge. Each rule in this representation corresponds to the use of a dispatching strategy subject to the existence of certain conditions. The basic language primitives are: Attribute, Atom, Predicate, Condition-Action rule, and the Rule composite.

Factory floor conditions are represented through various attributes that can take on nominal as well as continuous values (integer and real values). A function of Attributes also constitutes an Attribute. An atom, defined over an attribute, is a value restriction of the attribute. For example, an atom for a nominal attribute Priority may be (Priority = Urgent); and for a real valued attribute Processing-time, an atom may specify (5.0 <= Processing-time <= 8.5). Such atoms are referred to as Simple Atoms. Compound Atoms comprise of disjunctions of Simple Atoms.

Certain attributes may be job-related, in the sense that they may evaluate to different values for different jobs in the queue. The evaluation of an Atom of such an Attribute thus defines a subset of jobs whose values for the attribute are included in the atom's value-set.

A Predicate defines a function that selects from amongst a set of jobs. A Predicate may select a single job, or a subset of jobs. Standard dispatching rules like SPT, EDD, COVERT, etc. are modelled as predicates. Though simple dispatching rules do not select more than one job, Predicates retain the flexibility for representing other more complex algorithmic models that designate jobs for dispatch.

Traditional condition-action rules (called CA_Rules) are used to model various other scheduling heuristics. These may be strategies that human experts may have found useful. The condition part consists of conjunctions and/or disjunctions of Atoms, and the action part defines a Predicate.

Every Atom, Predicate and CA_Rule forms a Conjunct. A Rule consists of a conjunction of terms, with each term being a Conjunct (that is, either an Atom, a Predicate, or a CA_Rule). Every Conjunct can, in general, select a subset of jobs from the associated queue. Atoms that are not job-related can be considered as being True for all the jobs in the queue.

Each of the language constructs above define a Truth-value, which may take values from the set {True, False or Unknown}. An atom evaluates to True if the current value of its constituent Attribute is included in the values-subset defined by the Atom. If the current value of an Attribute cannot be determined, the corresponding Atoms return an

Unknown value. A Predicate evaluates to True if it selects a non-null subset of jobs from a queue. For a CA_Rule, its condition part is first evaluated, and if found to be True, the evaluation of the Predicate in the action part is returned as the CA_Rule's Truth-value. If any of the constituent Atoms of a CA_Rule do not evaluate to True, then the CA_Rule assumes the same Truth-value as that Atom.

In evaluating a rule, an ordering is assumed on the conjuncts forming the rule, such that all the Atoms are evaluated before the Predicates and the CA_Rules. As the conjuncts evaluate, each subsequent conjunct considers only the job subset selected by the prior evaluation.

The inferencing strategy using this rule language is described in Section 5.4.3. The representation in BNF is given below.

5.4.2 BNF

[ ]: any parameter listed in the square brackets is optional.

{ }: any parameter listed in the braces has to be specified.

truth_value: one of

    true false unknown

rule:

    AND (conjunct$_i$)

conjunct:

    atom

    predicate

    ca_rule

```
atom:

     simple_atom

     compound_atom

compound:

     OR(simple_atom_i)

simple_atom:

     (value_1 <= linear_attribute <= value_2)

     (nominal_attribute = value)

predicate:

     f:X→Y where Y⊆X

     Predicate is a function that selects a subset from a given set.

ca_rule:

     conditions

          logical_expression

     actions

          predicate

logical_expression:

     atom

     [logical_unary_opr]atom    [{logical_binary_opr}

                                 logical_expression]

logical_opr:

     logical_unary opr: one of
```

!(NOT)

logical_binary_opr: one of

&(AND) |(OR)

value:

number

symbol

attribute:

linear_attribute

nominal_attribute

function_of(attribute)

function_of(attribute):

$f: D \rightarrow X$ where D is the domain of the attribute and X is some nominal or continuous valued set.

5.4.3 Inference Strategy

A dispatcher's decisions are controlled by the inference scheme. The inference mechanism operates on the rules contained in the dispatcher's knowledge base, with the objective of selecting a single job for dispatch at each decision point. All jobs currently in the queue are possible candidates for dispatch.

In order to formalize the inference strategy, we first define the following:

Consider a rule R_j as:

R_j: C_{1j} AND C_{2j} AND ... AND C_{nj}. (Each C_{ij} here is called a conjunct.)

Definition: Rule R_j is *active* if all C_{ij} evaluate to True, i.e., if evaluate(C_{ij}) = TRUE for all i=1 to n.

Definition: Rule R_j is *consistent* if it is active and there is at least one common job voted for by all the conjuncts constituting R_j,

i.e., if evaluate(C_{ij}) = TRUE for all i=1 to n, and

AND (V_{ij}) != NULL, where V_{ij} = subset of jobs voted for by conjunct C_{ij}, and NULL is the null set.

Definition: A job can receive *actual votes* from any consistent rule. All common jobs voted for by all the conjuncts constituting the rule receive actual votes, i.e., all jobs included in AND(V_{ij}) get actual votes from that rule.

Definition: A job can receive *virtual votes* from any conjunct that evaluates to true. Note that even though Rule R_j may not be active, any Conjunct C_{ij} can give virtual votes to jobs as long as evaluate(C_{ij}) = TRUE.

Inference proceeds through the evaluation of all the rules in the knowledge base of the concerned dispatcher. In evaluating a rule, all the conjuncts in the rule are evaluated. If a conjunct evaluates to TRUE, it yields a set of jobs with virtual votes. A complete rule, if found consistent after evaluation, yields a set of jobs with actual votes. If an rule does not evaluate to be consistent, the sequence based evaluation procedure is overridden and its conjuncts assign virtiual votes to the job subset that the individual conjuncts select when evaluating independently.

At any particular decision point, the cumulative vote distribution is generated for the jobs waiting in the queue to be dispatched If an actual vote distribution exists then that is used for the purpose. If there are no consistent rules, that is, if there are no actual votes, then this vote distribution on the jobs is generated by using virtual votes. If there are no virtual votes, then a discrete uniform distribution is used as the job-vote distribution. After the generation of this cumulative vote distribution, a job is selected probabilistically in accordance with that distribution. Essentially, this policy seeks to ensure that the more the number of votes a job gets, the greater are the odds for it to be selected for processing.

Early experimentation with this voting scheme revealed a rather high degree of randomness in the job selection process. This happens when, at a decision point, there is more than one job with the same number of votes. In the absence of any conflict resolution policy, a job is then selected for dispatch at random.

An alternative is to use fitness based voting, whereby the votes assigned by rules and conjuncts are weighted by their respective current fitness values. As the results in Chapter 6 indicate, implementation of fitness based voting brings about significant performance improvements. A more sophisticated scheme would be to consider the variances of the fitness values, as in the variance sensitive bidding mechanism proposed in Goldberg (1990a). The weight associated with a rule or conjunct's votes would here be a function of its fitness value as well as its variance of the payoffs received, with higher variance interpreted as indicating lower reliability.

5.5 Genetic Learning for Scheduling

5.5.1 Recombination Operators

Four operators are defined for genetic recombination: Crossover (X), mutation (μ), Generalization (G) and Specialization (S). Crossover and Mutation are rule level operators, i.e., they operate on entire rules. Generalization and Specialization are defined to operate on certain conjuncts within rules. A conjunct level mutation operator is also defined. All these operators, with the exception of mutation, are based on the fitness-values of the conjuncts in the rules considered for recombination.

Rule-level operators

Crossover: This is an adaptation of the uniform crossover operator used with binary coded GAs (Syswerda, 1989). Given two parents, different Conjuncts are probabilistically selected for crossover based on their fitness values, with higher fitness Conjuncts having greater probability of participating in the crossover. For each Rule, the number of Conjuncts selected is approximately half the number Conjuncts in the rule. Since each selection is made probabilistically, based on the Conjuncts' fitness values, the same Conjunct may be selected more than once.

Consider the parents:

R_i : $C_{1i} \wedge C_{2i} \wedge C_{3i} \wedge C_{4i}$ (C_{2i} and C_{3i} are chosen for Crossover)

R_j : $C_{1j} \wedge C_{2j} \wedge C_{3j} \wedge C_{4j}$ (C_{1j} and C_{3j} chosen for Crossover)

Children may be formed in two ways:

(1) Child_i : $C_{1i} \wedge C_{1j} \wedge C_{3j} \wedge C_{4i}$

Child_j : $C_{2i} \wedge C_{2j} \wedge C_{3i} \wedge C_{4j}$

Here, Child_i is obtained from Rule_i, after replacing the selected conjuncts with those from the other parent. Child_j is similarly obtained.

(2) $\text{Child}_1 : C_{1j} \wedge C_{2i} \wedge C_{3i} \wedge C_{3j}$

$\text{Child}_2 : C_{1i} \wedge C_{2j} \wedge C_{4i} \wedge C_{4j}$

Here, Child_1 is formed by combining the chosen conjuncts from both parents. This rule thus now contains the "best" conjuncts from both parents. The second child, containing the weaker conjuncts, may be discarded.

Mutation: Mutation at the rule-level, occurs through the random replacing of one conjunct by another of the same type. Thus, if the Conjunct chosen for mutation is a Predicate or a CA_Rule, it is replaced by another randomly selected Predicate or CA_Rule respectively. In case the chosen Conjunct is an Atom, it is replaced by another randomly created Atom.

Conjunct-level operators: Recombination operators are also used to change the internal structure of the Conjuncts for Conjunct types suitable for such manipulation. Of the defined Conjunct types, only Atoms are considered for recombination. Predicates, being standard dispatching rules or other job-selection routines, do not have value assignments that may be used in recombination. CA_Rules too, are used to incorporate heuristics that a human dispatcher may use, and so no genetic operators are applied.

Let v be an attribute. Let $V = \{ v_1, ..., v_n \}$ be the set of values that it can take. Let a be an atom defined on that attribute. Let $A = \{ a_1, ..., a_k \}$, $A \subseteq V$, be the current binding of the atom a. Let $|V| = n$, and $|A| = k$.

Let U(1,k), $k \in \mathbb{R}$, be a discrete uniform distribution, that returns integers between 1 and k.

Let U(a,b), $a,b \in \mathbb{R}$, be a continuous uniform distribution, that returns real numbers between a and b.

From here on, if the attribute is nominal it will be assumed that v is a set of nominal valued elements, and so is the Atom a. If the attribute is continuous then it will be assumed that $v_i = [v_{i1}, v_{i2}]$ where v_{i1} , $v_{i2} \in \mathbb{R}$, and so is a_i.

Mutation (μ) is one of the random operators that is used to "jump start" the population. In the context of an Atom, mutation randomly selects an element in the current binding and alters it according to the type of the element. More formally it is defined as follows:

Let i = U(1,k), $k \in \mathbb{N}$, and let $a_i \in A$. Then, $\mu(a) = (A - \{a_i\}) \cup \{v_j\}$, where j = U(1,n) and $v_j \in V$.

Example 1: Let queue_length be an attribute. Let V = { [0,100]}. Let queue_length_q1 be an atom defined on queue_length. Let A = { [3, 7], [9, 11]}. If we consider this as a part of a production rule then it reads as

if ($3 \leq$ queue_length_q1 ≤ 7) $\vee$ ($9 \leq$ queue_length_q1 ≤ 11).

The Mutation operator μ(queue_length_q1) will pick one of the elements in A and alter it randomly by checking the domain of the Atom. Assuming that [3,5] is picked to be changed and that the new lower and upper bounds are 2 and 9. Then the resulting set A = { [2,9] } $\cup$ { [9,11] } = { [2,11] } .

Example 2: Let server_rate be an attribute. Let $V = \{100, 150, 200, 250\}$ and let server_rate_s1 be a nominal-valued Atom defined on server_rate. Let $A = \{100, 150\}$. Again as a production rule this reads as :

if (server_rate_s1 = 100) $\vee$ (server_rate_s1 = 150).

Then μ(server_rate_s1) will pick one of the elements in A and alter it according to the uniform distribution defined on the domain. Assume 150 is selected to be changed and assume 200 is selected from the domain. Then set $A = \{ 100 \} \cup \{ 200 \} = \{ 100, 200\}$.

<u>Generalization</u> (G) is performed when the strength of the atom is less then a specified threshold. Note that a low strength can occur if the atom is too specific to match any state seen so far. A generalization will relax the constraint imposed on the atom.

Let a be continuous, then $G(a) = A - \{a_i\} + v_i'$, where $v_i' \supset a_i$ and $v_i' \subseteq V$. If a is nominal, then $G(a) = A + \{v_i\}$, where $v_i \notin A$.

Example 3: Consider the same attribute of Example 1 of mutation. Then, G(queue_length_q1) will pick one of the elements and make its upper and lower bound strictly looser. The rest is exactly the same as in the mutation operator. In the nominal case, a new element from the domain will be added to the set A.

Contrary to the situation mentioned above, the strength of an atom can, at times, be too high. This is possible when the constraints on the atom are too loose, that is, it is too general. Such a situation will lead to over generalization which will, on the long run lead to a deterioration of the average system performance (that is, it will produce bad rules that fire all the time). In order to avoid such problems, the Specialization (S) operator is designed for use when the strength is higher than a particular threshold.

Let v be continuous. Then $S(a) = A - \{a_i + v'_i\}$, where $v'_i \subset a_i$. Let v be nominal, then $S(a) = A - \{a_i\}$, where $a_i \in A$.

Example 4: Considering the attribute mentioned in example 2 of mutation, S(server_rate_s1) will remove one of the elements in set A. In the continuous case, an element will be picked randomly and the upper and lower bounds will be made strictly tighter.

5.5.2 Credit Assignment

The credit assignment scheme currently implemented provides payoff to a job when it exits the system after completion of the last operation. This payoff value is transformed into a fitness values for all the Rules and Conjuncts that voted for the job. The payoff received for a job at the end of its operations is distributed across the knowledge bases of all the dispatchers that handled the job. In the case of Virtual Jobs, only that dispatcher that finally selected it is given credit.

The payoff criteria is modelled as an evaluation object, which may utilize a separate knowledge base for the purpose. A number of scheduling objectives, like minimization of flowtime, tardiness, maximization of utilization etc. are encoded in the system.

The fitness of a rule or conjunct is estimated as the average over the number of times it fired. The total accumulated fitness value is thus divided by the number of times a rule evaluated to be consistent, or, in the case of a conjunct, by the number of times the conjunct evaluated to TRUE.

Let $F(C_{ij})$ be the fitness of the ith Conjunct C_{ij} in $Rule_j$, and $F(R_j)$ the Overall Rule fitness.

We may define Overall Rule Fitness $F(R_j)$ as:

$$F(R_j) = W_1 F(C_j) + W_2 F(v)$$

where $F(v)$ is the Rule's average fitness obtained directly from the payoffs, and $F(C_j)$ is the contribution from the Conjuncts making up the Rule, which may be

$$F(R_j) = \min_i \{F(C_{ij})\}$$

or

$$F(R_j) = \sum_i F(C_{ij}) w_i$$

where w_i comes from a weight function, (weights varying inversely with $F(C_{ij})$).

A partial credit assignment mechanism is designed to provide credit to Conjuncts that evaluated to True, but where the Rules that they are a part of were not active. Since conjuncts may give Virtual Votes, even when the associated rule is not active, partial credit is assigned to individual conjuncts based on the Virtual Votes distribution. As seen from the definition of F (R_j), any increase in a Conjunct's fitness will also lead to an increase in the Rule's overall fitness. Here, since the rule was not active, the rule fitness $F(R_j)$ should not increase proportionately with the increase in any $F(C_{ij})$. The following compensation scheme is proposed:

The increase in a Conjunct C_{ij}'s fitness is given by

$$\Delta F(C_{ij}) = f(\text{virtual votes})$$

for some function f which may be defined as:

$$f_{obj}.k.\frac{\text{\# of virtual votes assigned by } C_{ij} \text{ to the selcted job}}{\text{total \# of virtual votes received by the selected job}}$$

where f_{obj} is the payoff received by the selected job. A possible set of values for k is

$$k = \begin{cases} .7 \text{ if } R_j \text{ is consistent} \\ .4 \text{ if } R_j \text{ is active but not consistent} \\ .2 \text{ if } R_j \text{ is not active.} \end{cases}$$

CHAPTER 6
JOB-SHOP SIMULATION

6.1 Overview

A framework for distributed decision support in a production scheduling environment using multiple intelligent objects has been proposed in Chapter 5. A scheme for genetic algorithm based learning of the requisite scheduling knowledge has been designed. This chapter examines the effectiveness of the developed facilities in the automated acquisition of scheduling strategies for a job-shop scheduling environment. The results of a number of simulation runs are analyzed to obtain preliminary insights into the sensitivity of the system to various parameters of the genetic learning scheme, and its adaptive ability in a dynamic scheduling situation.

The factory floor setup considered in the simulation experiments is the first nine stages from the front end of a semiconductor manufacturing line. For the purpose of this initial study, issues like wafer batch sizes and split batches have been ignored, and transportation times between tools have also not been taken into account. The performance of the learning system is compared against standard dispatching rules commonly used in scheduling decisions.

The system was developed using object-oriented programming techniques with C++, and the simulations were run on typical PS/2 hardware environment. Memory and processing speed proved to be a major limitation in conducting the experiments, thus limiting the number of manufacturing stages considered. With a single simulation run at times taking up to 48 hours and with limited hardware availability, it was also necessary to work with reduced learning periods and fewer number of learning episodes. Further, the simulation of the production facility with varying settings for the system parameters could only be conducted using single random number streams. The results presented in this chapter, therefore, provide only preliminary indications of the feasibility and capabilities of the developed genetic learning scheme for job-shop scheduling.

The simulation implementation and the shop floor setup considered are described in Section 6.2. The next section explains the different parameters of the developed system. Section 6.4 then presents the simulation results and discusses the behavior of the learning system.

6.2 Simulated Environment

6.2.1 Job Shop Setup

The manufacturing facility considered for the simulation study is a job shop environment, allowing alternate processing routes for a job. The processing sequence also involves looping back to certain machines (generally, testing facilities where jobs return after different operations). The production facility is organized around work-areas (or tool-numbers), each being occupied by multiple identical servers (or machines). Separate queues are present for different tool-numbers.

The simulated shop-floor setup is shown in Figure 6.1. Here, numbers inside the circles stand for the tool-numbers, and associated queues are indicated above the circles with numbers prefixed by a "Q". The number of servers for a particular tool-number are shown underneath the circles enclosed in square brackets. The arrows out of a tool-number represent the different possible tools that a job may be processed on in the next step, and the job undergoes processing in any one of these alternate tools. No preference ranking for processing amongst the alternative tools is considered.

As an example, with reference to Figure 5.1, one possible processing sequence for a job may be through the following tool-numbers: 1, 4, 2, 6, 2, 7, 1, 10, 2, 11. Another job may undergo processing along the route: 1, 3, 1, 6, 1, 7, 1, 9, 1, 11.

The processing times for a job at the different tools are given in Table 1. A setup time of .15 is considered for a change of operations on a machine. Machine breakdown and repair durations are calculated according to the exponential rates given in Table 2.

6.2.2 Implementation of the Simulation

An event oriented simulation model implemented in C++ using object oriented features is used for the experiments. The following object classes relate to the factory floor simulation: (see Figure 5.3 for relationships amongst object classes)

Node	Node objects are used to link the various different types of entities.
List	A set of linked nodes.
Event	Event objects, placed in the future events list, specify particular events to occur at specific times.
Queue	A list with some imposed ordering, and which can be associated with servers, dispatchers and other queues.
Server	Implements job processing facilities.

Table 1: Operation Processing Times

Operation Number	Processing Time
1	.06
2	.06
3	.46
4	.46
5	.46
6	.06
7	.06
8	.69
9	.06
10	.06
11	1.65
12	1.65
13	.06
14	.06
15	.15
16	.075
17	.06
18	.06
19	1.5

Table 2: Machine Breakdown and Repair Rates

Tool Number	Mean Time to Failure	Mean Time to Repair
1	19.0	1.0
2	19.0	1.0
3	17.28	4.32
4	17.38	4.32
5	15.0	5.0
6	13.0	7.0
7	13.0	7.0
8	13.0	7.0
9	19.0	1.0
10	19.0	1.0
11	13.0	7.0

Operations-List Specifies the sequence of operations to be performed on a particular job-type.

Job Jobs are linked their job-type related operations list.

Dispatcher A dispatcher associated with a particular queue is responsible for dispatch decision making. It carries a knowledge base that is used for selecting the next job from the queue for processing on any idle server associated with the queue.

Knowledge-Base Contains the rules used for the dispatch decision making, and has a genetic learning module for automated knowledge acquisition.

Evaluation Evaluation objects embody different evaluation criterion according to which jobs are evaluated and payoff assigned to knowledge bases that played a role in the dispatching of the job up to this particular point. Multiple evaluation points can be positioned at different points in the processing paths. The simulations presented in this chapter used single evaluations at the end of the job's processing sequence.

The queues, servers, and dispatchers are first created and the job-shop environment set up. The knowledge bases are initialized randomly. The simulation is initialized by inserting the following events into the future events list: the first job arrival, the termination of the simulation, the first breakdowns for all the machines/servers, and the first learning episode.

The simulation then proceeds through execution of the following events:

Arrival	A new job with associated attributes is created, and the next arrival is scheduled.
Setup-start	Initiated when a machine becomes idle and the queue is non-empty. A dispatch decision is made and the selected job is removed from the queue and linked to the machine. The job is removed from all virtual queues that it is currently contained in. The next event scheduled is a Setup-end or a Break-start, whichever occurs earlier. Relevant queue statistics are updated.
Setup-end	A Serve-start is scheduled and relevant statistics updated.
Serve-start	A Serve-end event is scheduled, and statistics updated.
Serve-end	The job is removed from the machine and placed in the next possible queues as virtual jobs. If there are no more processing to be performed on the job, a Leave event is scheduled. If the associated queue feeding into this machine is not empty, a Setup-start is scheduled. Relevant statistics are updated.
Break-start	The machine is flagged as busy if there is no job under processing. Otherwise, it is flagged as having failed with a job in service. A Break-end is scheduled. Relevant statistics are updated.
Break-end	A Serve-start is scheduled if a job was being processed when the breakdown occurred. Else, if the associated queue is non-empty,a Setup-start is scheduled. Relevant statistics are updated.

Learn	The genetic learning scheme is initiated for all knowledge bases.
Leave	The job leaves the system. A numeric payoff is estimated based on the evaluation criterion embodied in the evaluation object, and payoff is distributed in accordance with the credit assignment scheme. Relevant statistics are updated.
Terminate	Terminates the simulation after updating relevant statistics.

6.3 System Parameters

6.3.1 Job Stream

The initial set of experiments were conducted considering fixed processing times as given in Table 1. Jobs entered the system at an exponential arrival rate of .35, and 10% of the jobs were randomly assigned a high priority. Due dates for jobs were uniformly selected to be between 4 and 8 time units from the time the jobs entered the system.

A second set of experiments were conducted using variable processing times. For this, the processing time for an operation on a job was allowed to vary uniformly up to 50% about the values given in Table 1. For any particular job, the percentage variation about the given processing times was the same for all operations. Jobs arrived at an exponential arrival rate of .35, and 10% of the jobs were assigned high priority. Due dates here were uniformly chosen between 18 and 30 time units.

6.3.2 Learning System Parameters

The parameters of the learning system pertain to the genetic operators, the inference mechanism, and credit assignment policies. Details of the knowledge representation and genetic learning scheme have been discussed in Chapter 5. The different learning related parameters are defined below:

RFW1 and RFW2: A rule's fitness is the weighted average of the mean of its conjuncts fitness values, and the payoff received by the rule. RFW1and RFW2 are the weights assigned to the mean conjunct fitness and the rule payoff respectively.

MU: Rule level mutation probability.

CROSS: Crossover probability.

PGENSPEC: Probability of generalization or specialization of an atom.

THRESHG and THRESHS : Generalization and specialization thresholds.

PMUTATEC: Conjunct level mutation probability.

K_CONS, K_ACTIVE and K_INACTIVE: Constants used for partial credit assignment to conjuncts, based on whether the rule (that the conjunct is a part of) is consistent, active but not consistent, or is not active. (See Section 5.5.2)

LEARN_TIME: Time intervals at which a learning event is initiated. All results presented here used 1000 time units for a learning episode.

FIT_BASED_VOTES: Set to 1 if the votes given to jobs in a queue are to be weighted by the fitness values of the rules and conjuncts.

The fitness function for evaluating performance was modelled to incorporate a tradeoff between timely completion of priority jobs and minimization of overall job flow times. A single evaluation object was established at the end of the processing sequence, and a numerical payoff estimated according to the following rule:

```
For a priority job
    If a priority job completes processing on or ahead of schedule
        payoff = 10 / flowtime
    Else if a priority job completes processing after its due date
        payoff = 1 / flowtime
For a non-priority job
        payoff = 2 / flowtime.
```

6.3.3 Knowledge Representation Primitives

The knowledge representation primitives determine what rules (for dispatch decision making) can be formulated by the learning system. The following attributes, predicates, and condition-action rules were available to the genetic learning facility:

Attributes and their respective types (Nominal, Real or Integer valued):

Queue length of all queues (Integer)

Utilization of all machines(Real)

Mean time to failure of all machines (Real)

Expected service end time for all machines (Real)

Next queue length (Integer)

Critical ratio of a job (Real) calculated as the ratio of the remaining time till due of the job over its total remaining processing time.

Priority of a job (Nominal)

Time till due of a job (Real).

Predicates (dispatching rules):

First-Come-First-Serve (FCFS)

Last-In-First-Out (LIFO)

Shortest Processing Time (SPT)

Largest Processing Time (LPT)

Earliest Due Date (EDD)

Shortest Remaining Processing Time (SRPT)

Largest Remaining Processing Time (LRPT)

Fewest Remaining Operations (FRO)

Most Remaining Operations (MRO)

Least Critical Ratio (LCR)

Random (RNDM)

First-Come-First-Serve with Priority (FCPR)

Condition-Action Rules:

If cratio is between 0.8 and 0.1, then EDD.

If cratio is between 0.6 and 0.8, then LCR.

If cratio is between 1.0 and 2.0, then SPT.

If cratio is between -2.0 and 0.5, then EDD.

If priority is HIGH, then FIFO.

If priority is HIGH, then EDD.

If priority is HIGH, then SPT.

6.4 Simulation Results

In analyzing the results of the simulations, the following performance indicators are considered:

Average queue lengths of different queues

Total average flowtime of jobs through the system

Average tardiness of jobs

Average tardiness of priority jobs.

All averages are computed over the period of a learning episode. The graphs presented plot performance indicator versus the number of learning episodes.

Due to hardware related limitations, learning based dispatching was not implemented for all dispatchers. Preliminary experiments show low waiting times for jobs in queues Q3, Q4, Q5, Q9 and Q10 , and these queues were thus operated on a first-come-first-serve criterion. Queues, Q1 and Q2, though having low average waiting times too, were assigned "intelligent" dispatchers since jobs made five passes through them before completion of processing, and dispatch decisions made here could potentially impact performance.

The first set of simulations were conducted using fixed processing times as given in Table 1. Figures 6.2 through 6.5 compare the performance of the learning system with different values for RFW1 and RFW2, and first-come-first-serve is shown as the reference. The crossover and mutation rates here were 0.6 and 0.05 and K_CONS, K_ACTIVE and K_INACTIVE were set to 0.7, 0.5 and 0.3 respectively. All three parameter settings show improving performance as the number of learning episodes increase, with respect to average queue-lengths, and the average objective is also noted to dominate the values for FCFS. RFW1 and RFW2 set at .7/.3 is seen to yield the lowest average queue-lengths for all the shown queues.

Traditionally, binary coded genetic algorithms operate using low mutation rates of the order of .01 or lower. The performance of the developed scheme when using two different mutation rates of 0.15 and 0.2 is shown in Figures 6.6 through 6.8, and the higher mutation rate is seen to consistently perform better. The crossover rate here was 0.6, RFW1 and RFW2 were set equal at 0.5, and K_CONS, K_ACTIVE and K_INACTIVE were set to 0.7, 0.5 and 0.3 respectively.

The values set for K_CONS, K_ACTIVE and K_INACTIVE help differentiate between payoffs provided to conjuncts belonging to consistent, active but inconsistent, and inactive rules. An examination of the knowledge base during the learning process reveals the effect of these parameters. Consider the following two rules with associated fitness values:

Rule1: Rule-fitness = .005668

LPT, fitness = .001403

LPT, fitness = .001403.

Rule2: Rule-fitness = .003241

LPT, fitness = .000627

LPT, fitness = .000627

CA-Rule5, fitness = .000904.

Here, the higher fitness value of CA-Rule5 than that of LPT in Rule2 indicates that CA-

Rule5 performs better than LPT with respect to the performance objective. The fitness values for LPT are lower in Rule2 than in Rule1, possible because Rule 2 was not consistent and thereby its conjuncts get lower payoff when K_CONS is higher than the other two parameters. The lower conjunct fitnesses also make the overall rule fitness of Rule2 lower than that of Rule1. Rule1 thus has higher probability of being selection for reproduction and hence a better chance of surviving to the next generation, while the better performing CA-Rule5 does not get a chance to survive. A high performing conjunct, either entering the knowledge base through mutation and forming part of an inconsistent rule or through crossover finding itself as part of an inconsistent rule, can in this manner fail to survive.

This situation can be alleviated by assigning equal payoffs to conjuncts irrespective of whether they form part of a consistent, active but inconsistent, or inactive rule. The rest of the simulations thus use values of K_CONS, K_ACTIVE and K_INACTIVE set at 0.7.

Figures 6.9 through 6.14 show at comparison of the performance of the learning system with FCFS, SPT and EDD. In the latter cases, the dispatching decisions at all queues were made in accordance with the respective dispatching rule. The poor performance of SPT here can be explained by noting that since fixed processing times are used, all jobs are optimal with respect to SPT, and the performance of SPT is thus essentially random. The graphs show that the learning system overall yields lower average queue lengths. As Figures 6.13 and 6.14 indicate, learning also results in smaller peaks in the average flowtime and tardiness plots when compared with single dispatching rules.

An examination of the knowledge bases in the above simulations also revealed a high degree of randomness in the dispatch decision, resulting from the voting mechanism employed. In the voting scheme used, all rules and conjuncts assigned a single vote to chosen jobs irrespective of their fitness values or any other criterion. Any conflicts were thus resolved randomly. If, for example, a knowledge base contained five consistent rules with each voting for a different job, then one out of these five jobs would be finally selected for dispatch at random. A fitness based voting scheme was therefore implemented to overcome this shortcoming. Noting that high fitness rules and conjuncts, in view of their better performance, deserve more importance in the decision making, votes given by the rules and conjuncts are weighted by their current fitness values.

A second factor that contributed to the randomness in job selection was noticed to arise from the knowledge base at times degenerating to a set of non-discriminating rules. This happens for non-job related atoms when they evaluate to true, and predicates that do not discriminate between jobs but instead vote for all jobs waiting in the queue. In this situation, irrespective of the particular job dispatched, all such non-discriminating atoms and predicates receive the payoff associated with this job. These conjuncts thus acquire fitness values at par with or even higher than the best performing conjuncts present, and with this rapidly accumulating strength soon begin to dominate. When all conjuncts are of this nondiscriminating type, the knowledge base is degenerate, selecting jobs for dispatch at random.

For the remaining simulations, this degeneracy is sought to be controlled by removing nondiscriminating dispatching rules and considering a minimal number of non-job related atoms. The FRO and MRO dispatching rules, which can differentiate between jobs only in queues Q1 and Q2, are not considered. The server related attributes and most queue length attributes are also removed from consideration for the next set of simulations. Though this reduced the problem. the knowledge base was still noticed to degenerate at times.

The second set of experiments considered variable processing times as described in Section 6.3.1. The evaluation criterion was the same as outlined in Section 6.3.1. The due date range used is higher than in the earlier experiments, resulting in fewer jobs completing behind schedule.

A comparison of fitness based voting with the earlier scheme is presented in Figures 6.15 through 6.21. As seen from these graphs, conflict resolution through fitness based voting brings about significant benefits in all performance indicators. For these experiments, RFW1, RFW2, and the crossover and mutation rates were set at 0.3, 0.7, 0.6 and 0.2 respectively.

Figures 6.22 through 6.27 compare different values for RFW1 and RFW2. The 0.3/0.7 setting for RFW1/RFW2, which provides higher weightage to the payoff received directly by consistent rules, in calculating a rules overall fitness, is noticed to have an edge over the other values. RFW1 and RFW2 are thus set at 0.3 and 0.7 for the rest of the simulations.

A comparison of the system performance under different mutation probabilities is shown in Figures 6.28 to 6.32. Low mutation rates result in more stable knowledge bases, while higher values can cause random disturbances in the evolving rules. With the used knowledge representation, however, access to primitives not currently in the knowledge base can result only from mutation. The designed crossover operator can only interchange the conjuncts in different rules and cannot bring in new conjuncts into the knowledge base. With the system being thus dependent, to a large extent, on mutation to evolve new conjuncts, higher mutation probability (higher than that typical in the case of binary string encodings) are necessary. The mutation probability set at 0.2 is noted to yield best overall performance. Increasing this to 0.3, however, is noticed to be detrimental. High mutation probability can bring about a degenerating knowledge base resulting from some mutation induced changes, and can also eliminate a high performing conjunct before allowing it to acquire high fitness and evolve. The low mutation rate of 0.05 is seen to perform well in Figure 6.28, but exhibits worsening performance in Figure 6.30.

The effect of different crossover probabilities on the system performance is given in Figures 6.33 through 6.37. Distinctly higher queue lengths are observed for the crossover rate set at 0.7 in Figures 6.34 to 6.35. A higher crossover rate implies that fewer rules will survive intact to the next generation, and crucial high performing rules may thus be disrupted before being allowed an opportunity to dominate the knowledge base.

The performance of the learning system is compared with that of standard dispatching rules in Figures 6.38 through 6.48. The dispatching rules considered are FCFS, SPT, EDD, FCPR and RNDM.

Figures 6.38 to 6.41 show that the learning system evolves to perform better than all the above dispatching rules, with large differences observed between the average queue lengths of the learning and non-learning based simulations in Figures 6.39 to 6.40.

The overall average flow times are displayed in Figures 6.42, and learning is observed to result in lower peaks than the other dispatching rules. In order to clearly illustrate the differences, a comparison of learning versus SPT and FCPR is shown in Figures 6.43 and 6.44. Though SPT is generally expected to perform well with respect to average flowtime criterion, Figure 6.43 shows that the learning system is able to avoid the high peaks that SPT exhibits. In Figure 6.44, learning is also seen to perform better than FCPR , a dispatching rule used often in practice.

The overall average tardiness of jobs is compared in Figure 6.45, and again learning is noticed to result in lower peaks than single dispatching rules. It is to be noted here that no explicit tardiness related criterion was incorporated into the evaluation function, except for priority jobs producing higher payoffs when they complete processing on or ahead of their due dates. A comparison of the tardiness of priority jobs with FCFS and FCPR is presented in Figures 6.46 and 6.47. Even better tardiness based performance can be expected with some tardiness related measure as part of the evaluation criterion.

Simulations of the job shop environment is thus seen to indicate improved scheduling performance with the use of the developed genetic learning scheme. The experiments conducted with the developed system, however, reveal certain shortcomings requiring enhancements to the current facilities. These together with other future research issues are discussed in Chapter 8.

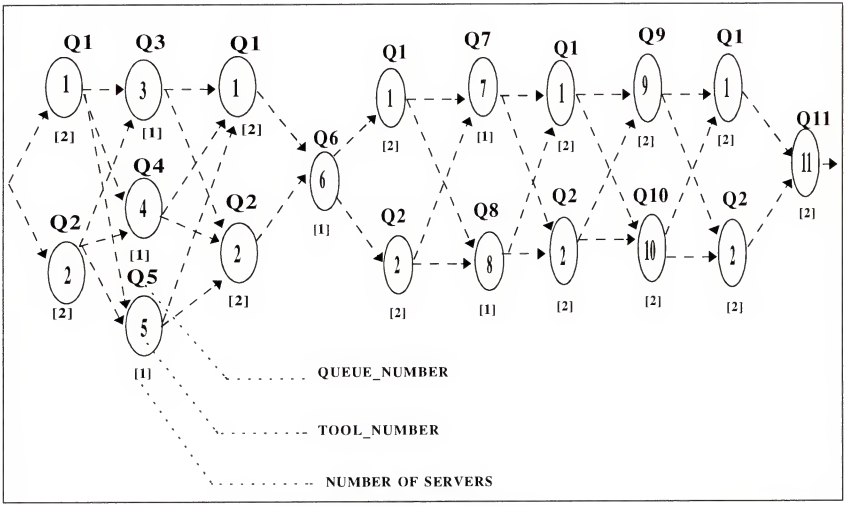

Figure 6.1: Jobshop Setup

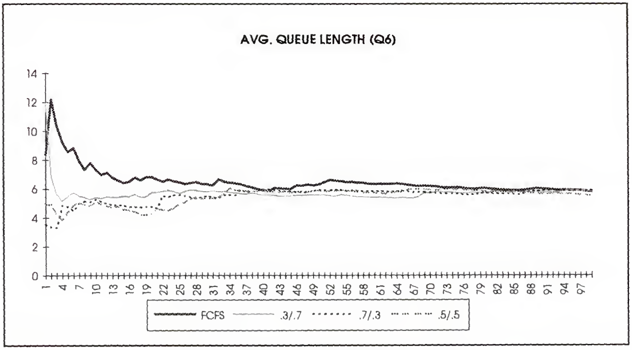

Figure 6.2: Comparison of Rule Fitness Weights
(Fixed Processing Time Experiments)

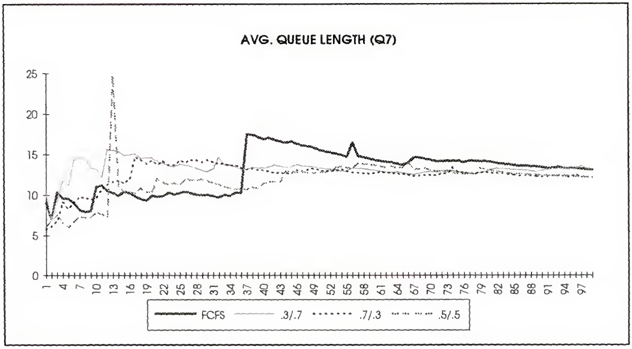

Figure 6.3: Comparison of Rule Fitness Weights
(Fixed Processing Time Experiments)

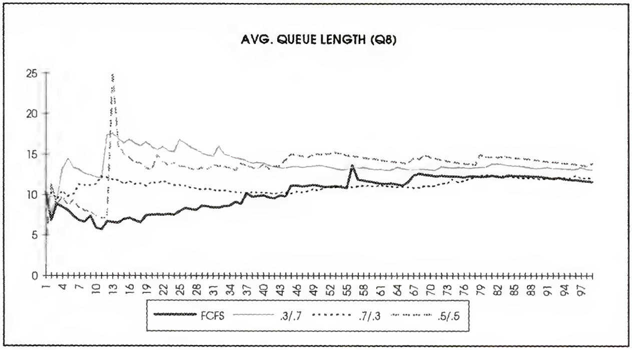

Figure 6.4: Comparison of Rule Fitness Weights
(Fixed Processing Time Experiments)

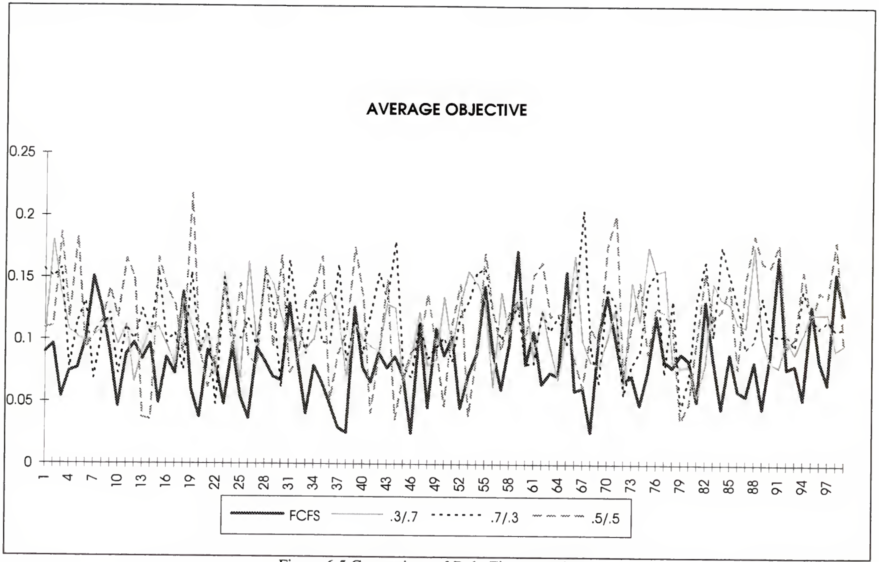

Figure 6.5:Comparison of Rule Fitness Weights
(Fixed Processing Time Experiments)

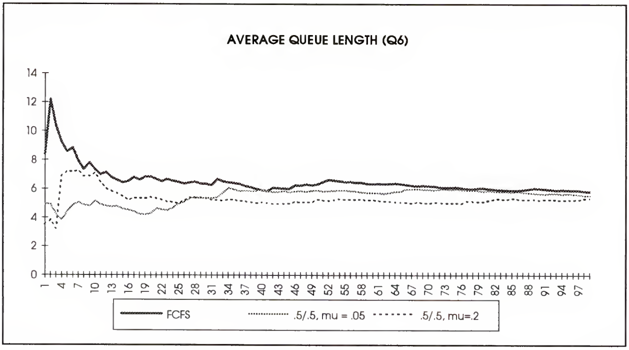

Figure 6.6: Comparison of Mutation Rates
(Fixed Processing Time Experiments)

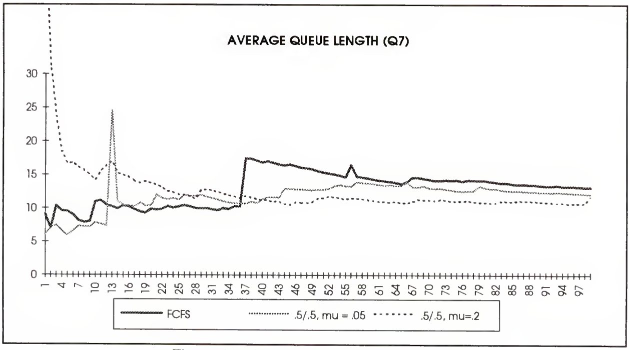

Figure 6.7: Comparison of Mutation Rates
(Fixed Processing Time Experiments)

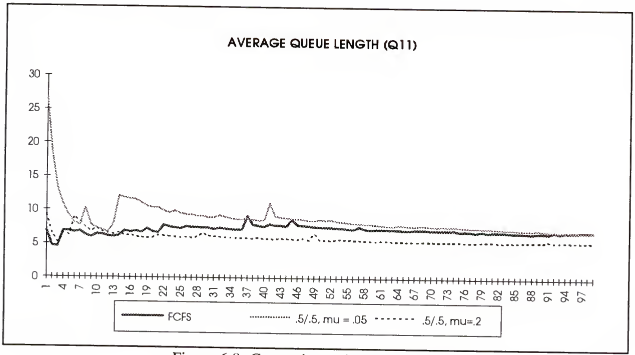

Figure 6.8: Comparison of Mutation Rates
(Fixed Processing Time Experiments)

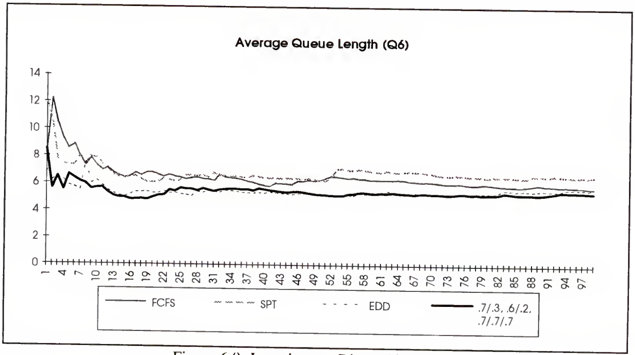

Figure 6.9: Learning vs. Dispatching Rules
(Fixed Processing Time Experiments)

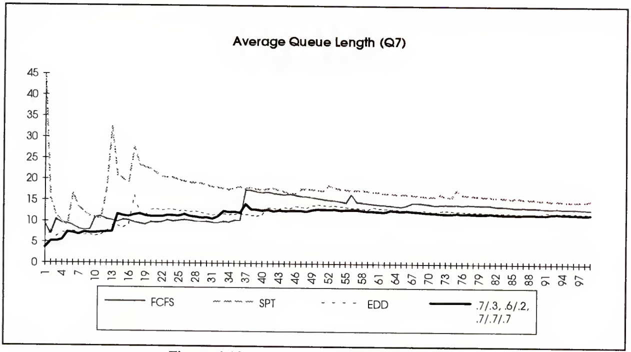

Figure 6.10: Learning vs. Dispatching Rules
(Fixed Processing Time Experiments)

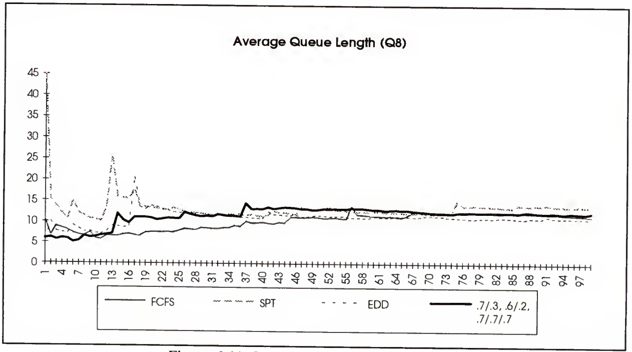

Figure 6.11: Learning vs. Dispatching Rules
(Fixed Processing Time Experiments)

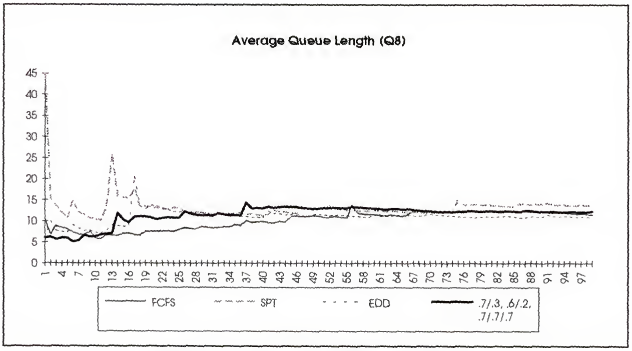

Figure 6.12: Learning vs. Dispatching Rules
(Fixed Processing Time Experiments)

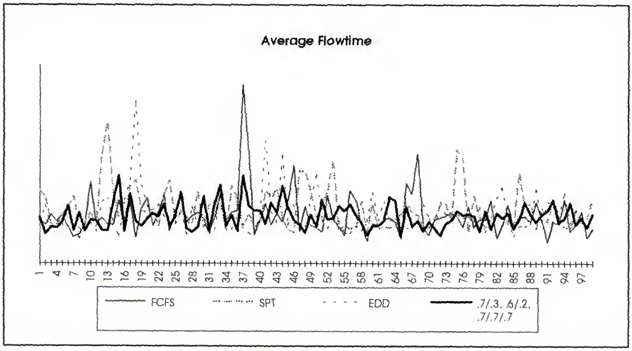

Figure 6.13: Learning vs. Dispatching Rules
(Fixed Processing Time Experiments)

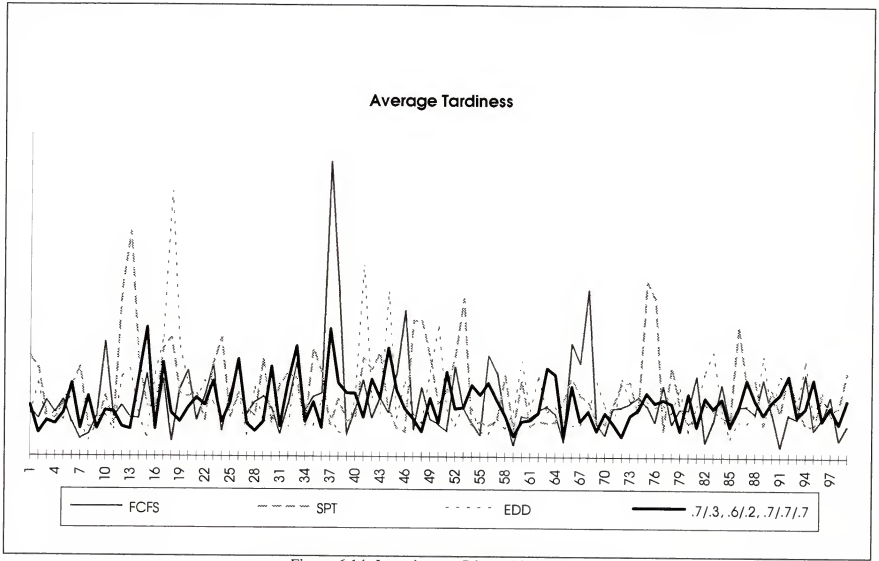

Figure 6.14: Learning vs. Dispatching Rules
(Fixed Processing Time Experiments)

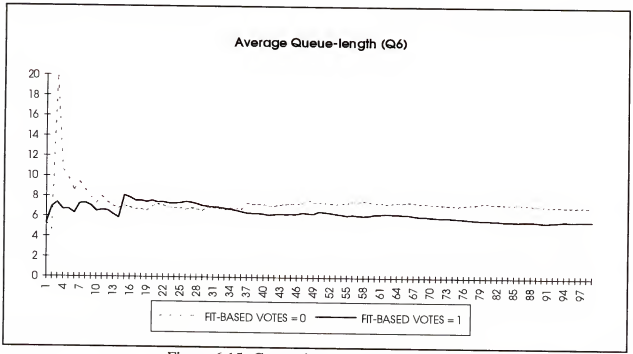

Figure 6.15: Comparison of Voting Schemes

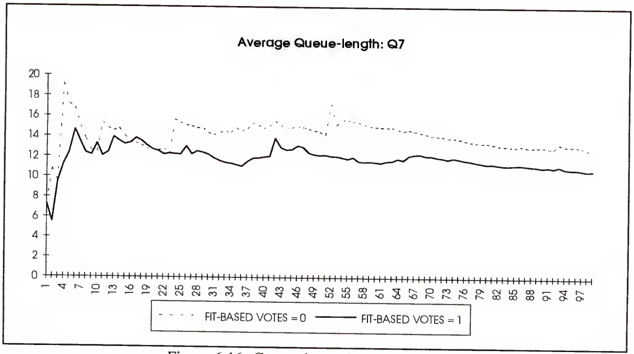

Figure 6.16: Comparison of Voting Schemes

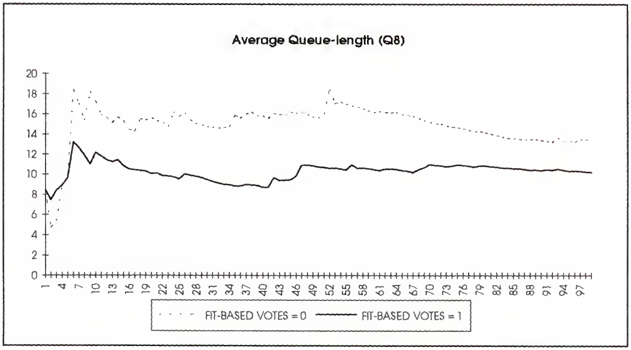

Figure 6.17: Comparison of Voting Schemes

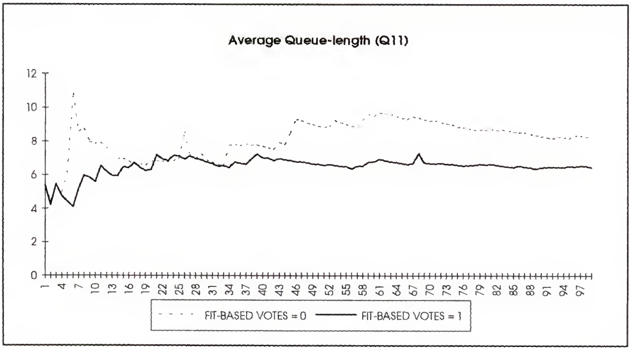

Figure 6.18: Comparison of Voting Schemes

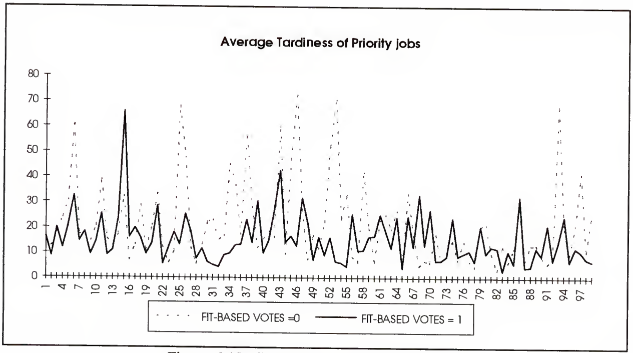

Figure 6.19: Comparison of Voting Schemes

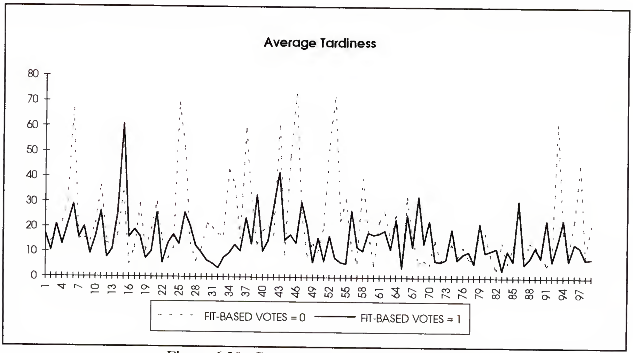

Figure 6.20: Comparison of Voting Schemes

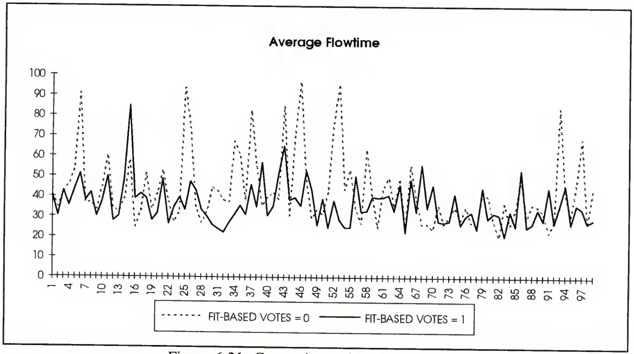

Figure 6.21: Comparison of Voting Schemes

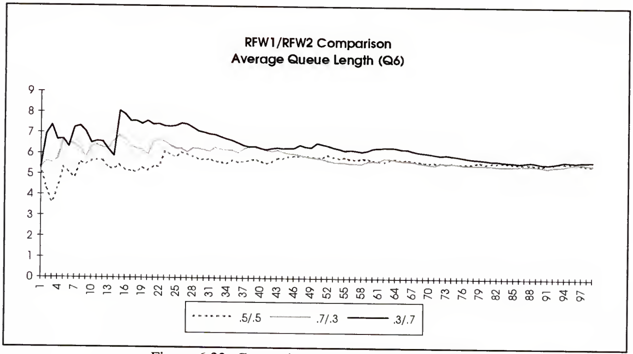

Figure 6.22: Comparison of Rule Fitness Weights

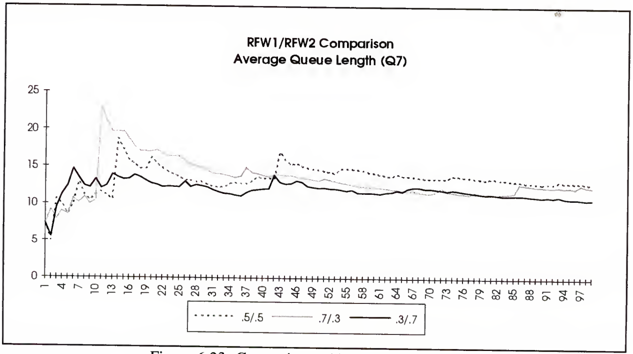

Figure 6.23: Comparison of Rule Fitness Weights

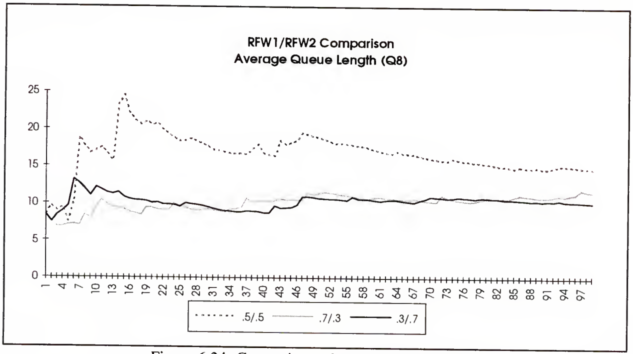

Figure 6.24: Comparison of Rule Fitness Weights

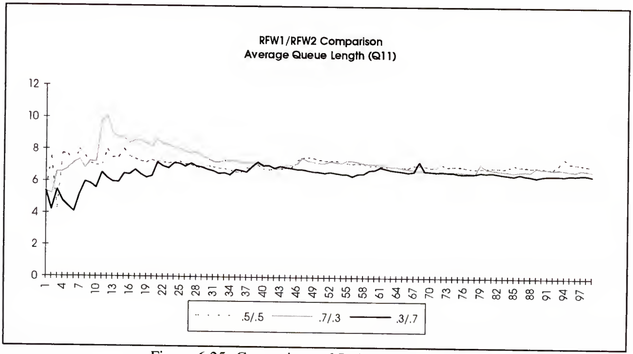

Figure 6.25: Comparison of Rule Fitness Weights

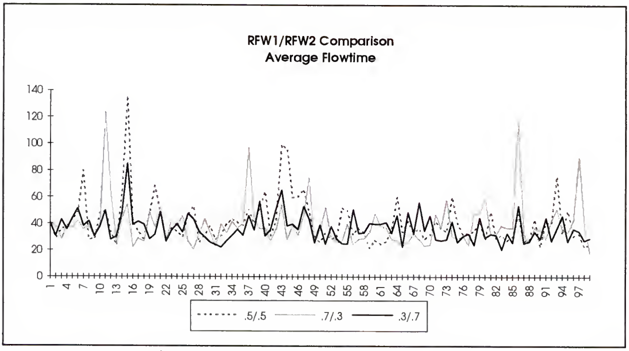

Figure 6.26: Comparison of Rule Fitness Weights

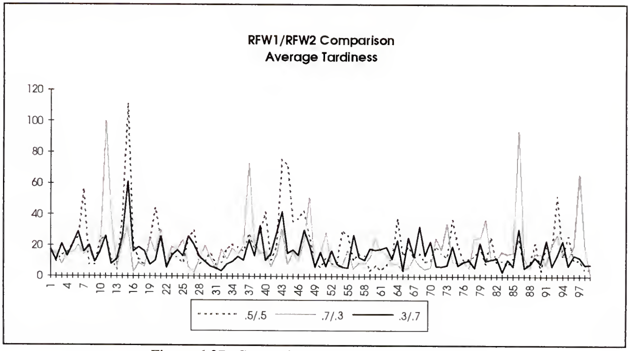

Figure 6.27: Comparison of Rule Fitness Weights

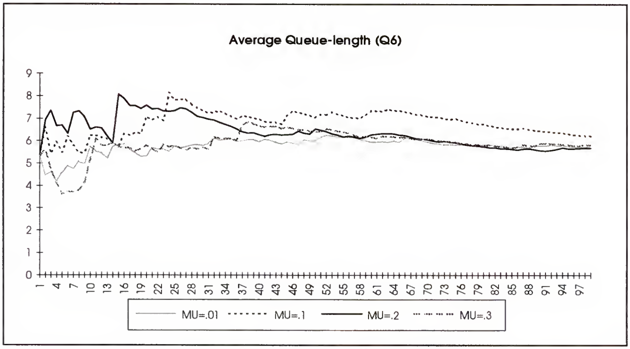

Figure 6.28: Comparison of Mutation Rates

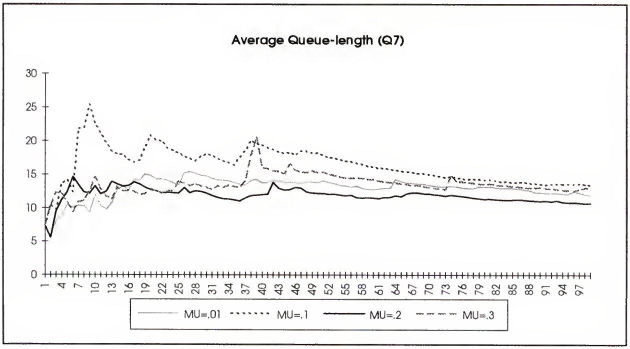

Figure 6.29: Comparison of Mutation Rates

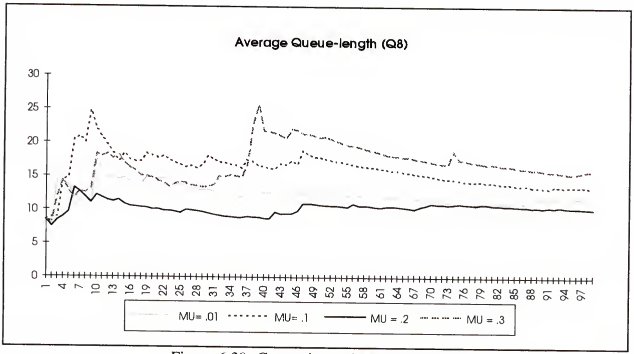

Figure 6.30: Comparison of Mutation Rates

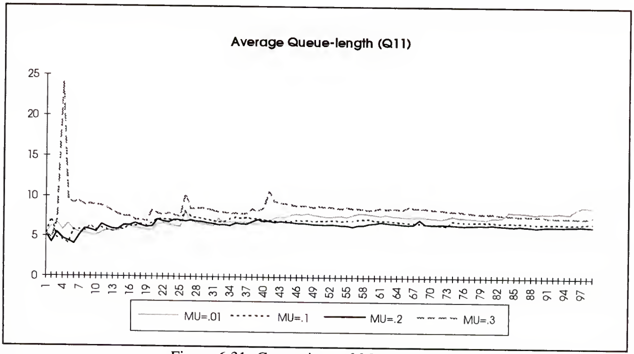

Figure 6.31: Comparison of Mutation Rates

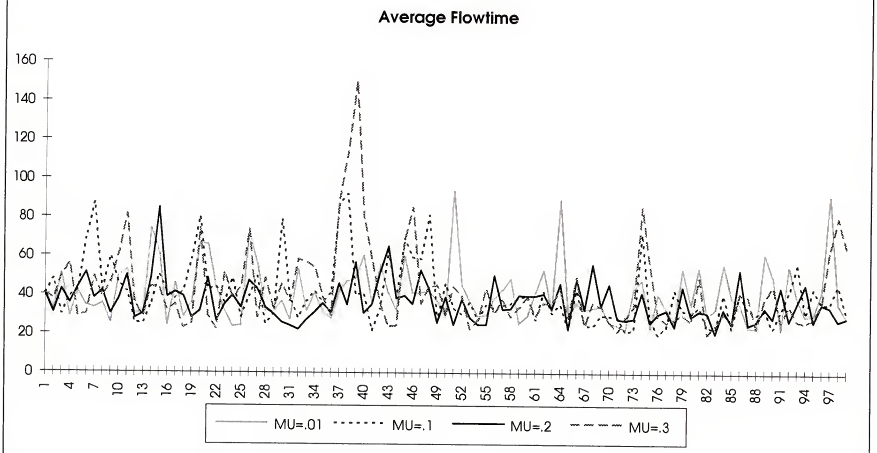

Figure 6.32: Comparison of Mutation Rates

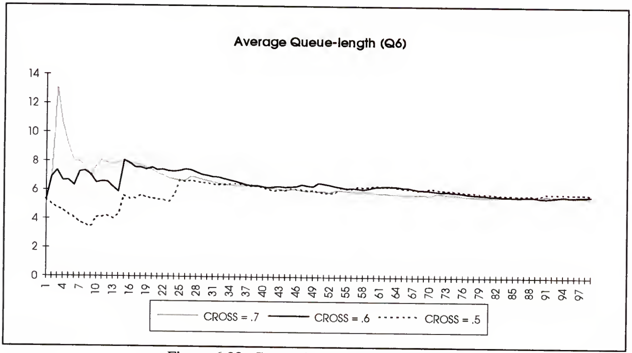

Figure 6.33: Comparison of Crossover Rates

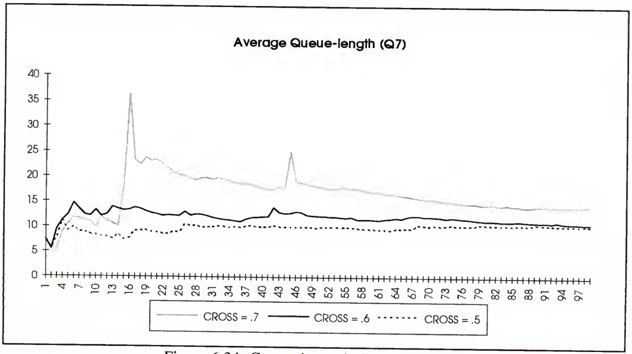

Figure 6.34: Comparison of Crossover Rates

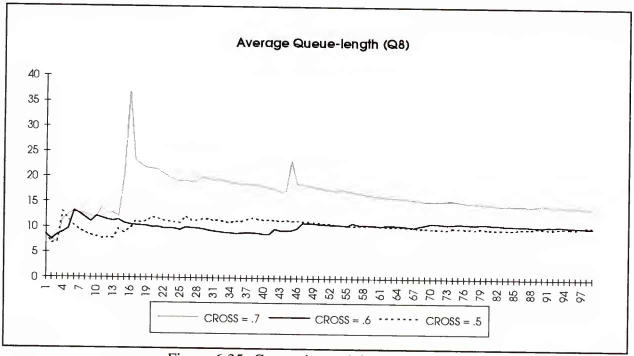

Figure 6.35: Comparison of Crossover Rates

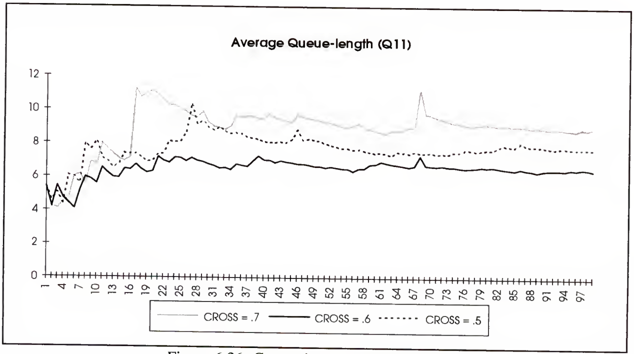

Figure 6.36: Comparison of Crossover Rates

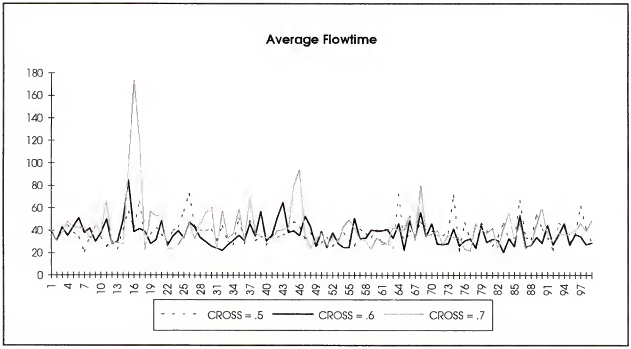

Figure 6.37: Comparison of Crossover Rates

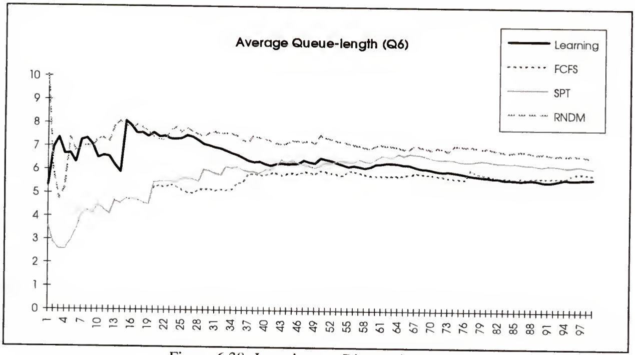

Figure 6.38: Learning vs. Dispatching Rules

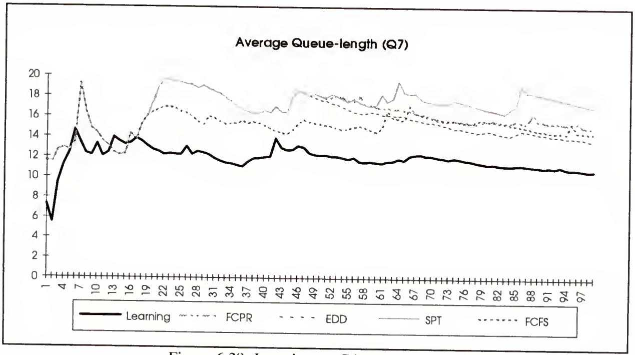

Figure 6.39: Learning vs. Dispatching Rules

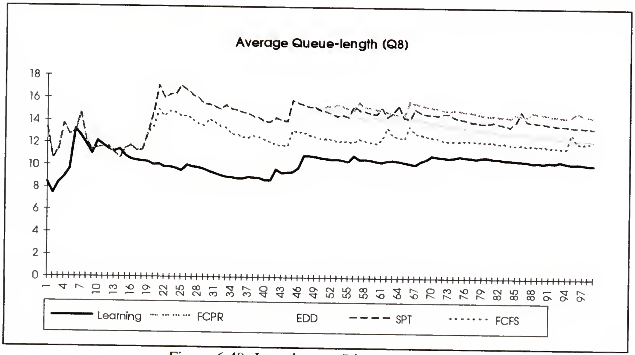

Figure 6.40: Learning vs. Dispatching Rules

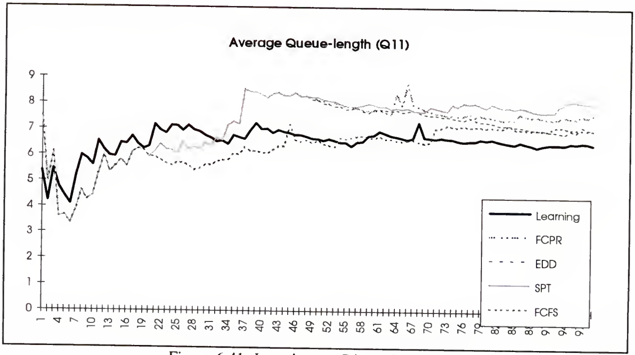

Figure 6.41: Learning vs. Dispatching Rules

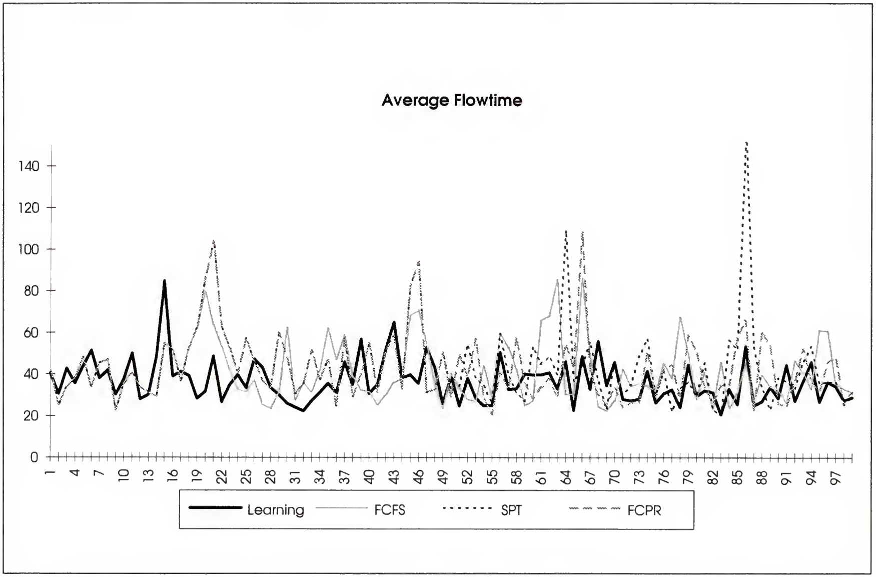

Figure 6.42: Learning vs. Dispatching Rules

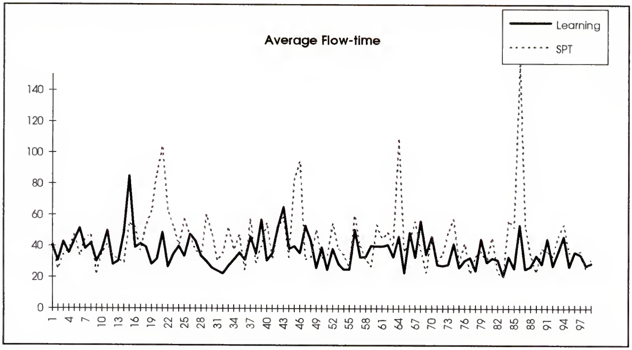

Figure 6.43: Learning vs. Dispatching Rules

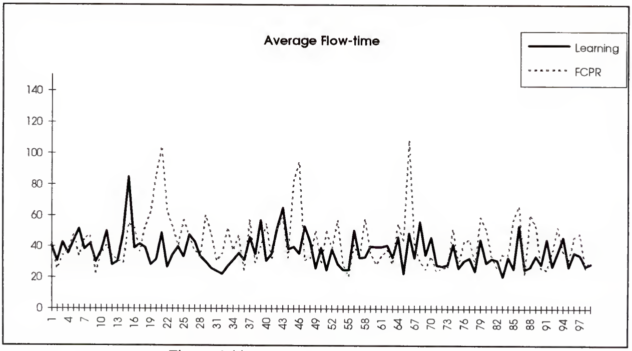

Figure 6.44: Learning vs. Dispatching Rules

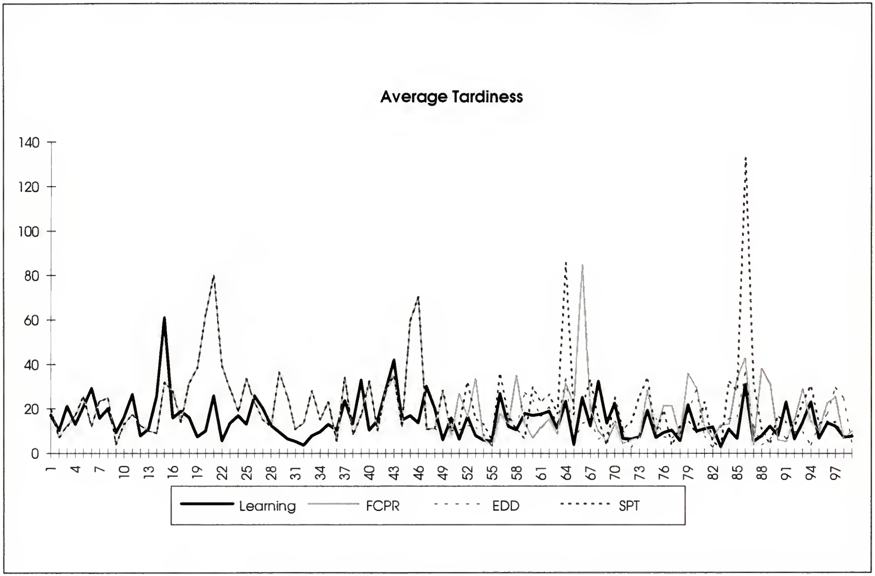

Figure 6.45: Learning vs. Dispatching Rules

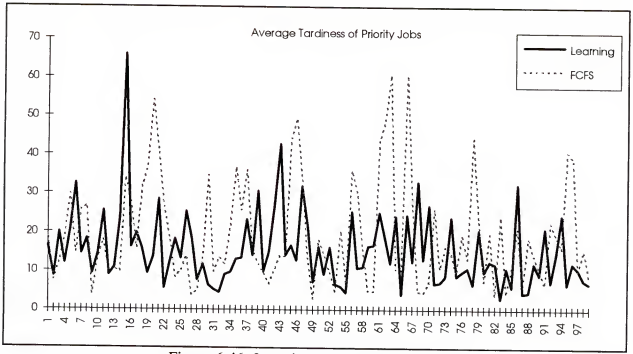

Figure 6.46: Learning vs. Dispatching Rules

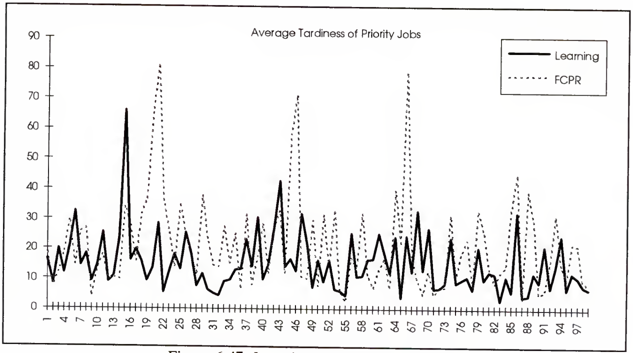

Figure 6.47: Learning vs. Dispatching Rules

CHAPTER 7
GENERALIZATION OF THE VOSE-LIEPINS FRAMEWORK

7.1 Introduction and Notation

The mathematical framework provided by Vose and Liepins (1991) for analyzing the behaviour of a Simple Genetic Algorithm has been desribed in Section 4.4. There, a binary representation scheme was assumed. In this chapter, an extension of this formalism is developed for Genetic Algorithms using a nonbinary representation. Simple genetic search, using the traditional one-point crossover and uniformly random mutation on fixed length strings using a high cardinality alphabet is considered.

A key result of the Vose-Liepins model was based on a conjecture obtained from computational results. This conjectured result has, since, been proved (Koehler, 1992). In this chapter, analogous results for the generalized case of nonbinary alphabets, are provided. Walsh transforms have formed an important part in the theoretical analysis of GAs, and the above framework makes extensive use of Walsh matrices. The results here are also based on an extension of the Walsh matrix definition to cover nonbinary representations.

A representation based on an alphabet of cardinality K is considered. In order to preserve the basic mechanics of the Vose-Liepins framework, and to allow the use of the Walsh matrix, K is taken to be some power of 2. Though this assumption restricts generalizability to alphabets of arbitrary cardinality, the results nonetheless allow a comparison of properties of binary and nonbinary GSa and brings us a step closer towards a theoretical basis for analyzing nonbinary GAs.

After defining the notation below, the generalized Walsh matrix terms are obtained in Section 7.2. Then, in Section 7.3, the recombination operator expression is derived. Properties related to the stability of the fixed points of M are given in Section 7.4.. The expression for the spectrum of M is then derived in Section 7.5.

The following notation is used throughout.

<u>Notation:</u>

K	is the cardinality of the alphabet; $K = 2^v$, for some integer v.
L	is the length of the nonbinary string.
χ	is the crossover rate.
μ	is the mutation rate.
$d_i(s)$	is the ith digit of the string s.
$\delta(d_i(s))$	is 1 if the ith digit of s is non-zero, and is 0 otherwise.
$\sum_{i=p}^{q} \delta(d_i(s))$	is the number of non-zero digits in s from the pth to the qth positions.
$\lvert s \rvert$	is the total number of non-ze ~~nonbinary~~ digits in the string s.
rev_v (s)	is the string obtained by taking the digits of s in reverse order.
wid(s)	is the defining length of the string s, that is, it is the number of digits in s between the outermost non-zero digits; wid(0) = 0.
del(x)	is 1 if x = 0, and is 0 otherwise.
b(s)	is the binary equivalent of the nonbinary string s, obtained by

concatenating the v-bit binary representation for every digit in s.

e is a vector of ones of appropriate length.

7.2 Generalization of Walsh Matrix Terms

A direct method for computing the Walsh matrix terms in the binary case has been provided in Koehler (1992):

Proposition 1 :

For binary strings i and j, the Walsh matrix terms are given by:

$$W_{i,j} = (-1)^{|rev(i) \otimes j|} = (-1)^{|rev(j) \otimes i|}.$$

Considering nonbinary strings r and s, we have

$$W_{r,s} = (-1)^{|b(r) \otimes b(rev_v(s))|}.$$

Representing the strings by their individual digits, we get

$$= (-1)^{|b(\sum_{i=1}^{L} d_i(r)K^{i-1}) \otimes rev_v(b((\sum_{i=1}^{L} d_i(s)K^{i-1})|}.$$

However, the second term in the exponent is

$$rev_v(b(\sum_{i=1}^{L} d_i(s)K^{i-1}) = \sum_{i=1}^{L} K^{L-i} rev_v(b(d_i(s))).$$

Switching the second index size gives

$$\sum_{i=1}^{L} K^{L-i} \mathrm{rev}_v b(d_i(s)) = \sum_{i=1}^{L} K^{i-1} \mathrm{rev}_v b(d_{L+1-i}(s)).$$

Thus the Walsh terms may be written as

$$W_{r,s} = (-1)^{\left| b\left(\sum_{i=1}^{L} d_i(r) K^{i-1} \right) \otimes \sum_{i=1}^{L} K^{i-1} \mathrm{rev}_v(b(d_{L+1-i}(s))) \right|}$$

$$= (-1)^{\left| \sum_{i=1}^{L} K^{i-1} (b(d_i(r)) \otimes \mathrm{rev}_v(b(d_{L+1-i}(s))) \right|}.$$

Since the expression in $|.|$ cannot have negative terms in the sum, we may write

$$W_{r,s} = (-1)^{\sum_{i=1}^{L} | d_i(r) \otimes \mathrm{rev}_v(d_{L+1-i}(s)) |}$$

$$= \prod_{i=1}^{L} W_{d_i(r),\, d_{L+1-i}(s)}$$

$$= \prod_{i=1}^{L} W_{d_i(r),\, d_i(\mathrm{rev}_v(s))}.$$

We thus obtain the following theorem:

Theorem 1: The Walsh martix terms $W_{r,s}$ for nonbinary strings r and s of L digits and cardinality $K = 2^v$ are given by

$$W_{r,s} = \prod_{i=1}^{L} W_{d_i(r),\, d_{L+1-i}(s)} = \prod_{i=1}^{L} W_{d_i(r),\, d_i(rev_v(s))} \; .$$

Note that d_i (r) above respresent the ith symbol (digit) of the string r. Considering an alphabet of cardinality K, this term is obtained from the KxK Walsh matrix, each term of which is given by Proposition 1 (where the binary equivalent of r is used).

7.3 The Recombination Operator

Considering recombination through simple crossover and mutation, the following relationship has been derived (Vose and Liepins (1991):

$$r_{i,j}(k \oplus l) = r_{i \oplus k,\, j \oplus k}(l) \, .$$

The probabiliity $r_{i,j}(k)$ of any two binary strings i and j yielding a string k can thus be obtained as

$$r_{i,j}(k) = r_{i,j}(k \oplus 0) = r_{i \oplus k,\, j \oplus k}(0)$$

and so we only need to be able to compute $r_{i,j}(0)$ in order to get all the recombination probabilities. We note that in the higher cardinality alphabet case too, this same relationship is seen to hold by considering binary versions of the strings.

The probabliilties $r_{i,j}(0)$ for the nonbinary case are derived below. The mutation probability on a string r is given by

$$m(r) = \prod_{i=1}^{L} [\ \delta(d_i(r)) \frac{\mu}{K-1} + (1-\delta(d_i(r)))(1-\mu)\ ]$$

$$= (\frac{\mu}{K-1})^{|r|} (1-\mu)^{L-|r|}$$

$$= (\frac{\mu}{1-\mu})^{|r|} \frac{(1-\mu)^L}{(K-1)^{|r|}}$$

$$= (\frac{\eta}{K-1})^{|r|} (1-\mu)^L.$$

Now for single point Crossover, let p be the randomly chosen crossover point. Consider two parent strings s and t, and let c1 and c2 be the two children resulting from crossover. Then, the number of non-zero digits in c1 and c2 are given by

$$|c1| = |s| - \sum_{i=1}^{p} \delta(d_i(s)) + \sum_{i=1}^{p} \delta(d_i(t))$$

and $$|c2| = |t| - \sum_{i=1}^{p} \delta(d_i(t)) + \sum_{i=1}^{p} \delta(d_i(s)).$$

Defining

$$\Delta_{s,t,p} = \sum_{i=1}^{p} \delta(d_i(s)) - \sum_{i=1}^{p} \delta(d_i(t)),$$

we may write

$$|c1| = |s| - \Delta_{s,t,p}$$

and $$|c2| = |t| + \Delta_{s,t,p}.$$

Then, considering mutation and crossover together, the probability of c1 being the string 0 is

$$(1-\chi)\left(\frac{\eta}{K-1}\right)^{|s|}(1-\mu)^L$$

if no crossover actually occurs (with probability (1-χ)), and it is

$$\frac{\chi}{L-1}\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|s|-\Delta_{s,t,p}}(1-\mu)^L$$

if crossover does take place. Thus, the probablilty that the child c1 is the string 0 is given by

$$(1-\chi)\left(\frac{\eta}{K-1}\right)^{|s|}(1-\mu)^L + \frac{\chi}{L-1}\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|s|-\Delta_{s,t,p}}(1-\mu)^L .$$

Similarly, the probability of c2 being 0 is

$$(1-\chi)\left(\frac{\eta}{K-1}\right)^{|t|}(1-\mu)^L + \frac{\chi}{L-1}\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|t|+\Delta_{s,t,p}}(1-\mu)^L .$$

Since, c1 and c2 can be the string 0 with equal probability, we have the following result:

Theorem 2: The probability of any two strings s and t recombining, through single point crossover and uniformly random mutation, to yield the string 0 is

$$r_{s,t}(0) = \frac{1}{2}\left[(1-\chi)\left(\frac{\eta}{K-1}\right)^{|s|}(1-\mu)^L + \frac{\chi}{L-1}\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|s|-\Delta_{s,t,p}}(1-\mu)^L\right]$$

$$+ \frac{1}{2}[(1-\chi)(\frac{\eta}{K-1})^{|t|}(1-\mu)^L + \frac{\chi}{L-1}\sum_{p=1}^{L-1}(\frac{\eta}{K-1})^{|t|+\Delta_{s,t,p}}(1-\mu)^L].$$

Vose and Liepins (1991) have formalized simple genetic recombination through an operator $\bar{M}$ described in Section 4.4. A related matrix, the twist of M, denoted M*, plays an important role their analyses, and is defined below:

Proposition 2: (Definition and Properties of M*)

1. $M^*_{i,j} = M_{i \oplus j,\, i}$
2. W M* W is lower triangular.

The second property has been proved for binary strings. In the nonbinary case, since K is assumed to be some power of 2, we may convert any nonbinary string to its binary equivalent. This relation is, then, also true for the nonbinary strings considered here.

7.4 Stability of Fixed Points

The stability of the fixed points of the recombination operator $\bar{M}$ has been studied, and the following result relates stability to the eigenvalues of M*:

Proposition 3: (Vose and Liepins, 1991)

If the matrix M is positive, then any fixed point of $\bar{M}$ is stable whenever the second largest eigenvalue of M* is less than 1/2.

Based on computational results, the authors had conjectured that the eigenvalue of M* would be less than 1/2 if the mutation rate is kept between 0 and .5. An expression for the spectrum of M* has been obtained in Koehler (1992). In this paper we derive the spectrum of M* for the nonbinary case.

7.5 Spectrum of M*

7.5.1 Preliminary Expression

In obtaining the spectrum of M*, we make use of the following results derivable from Proposition 2.2 above :

Lemma 1:

Let $C = W M^* W$.

1. Then, the eigenvalues of M^* are $\frac{C_{i,i}}{K^L}$, for $i = 0,\dots,K^L$.

2. The $C_{i,i}$ values are given by

$$C_{i,i} = \sum_{i=0}^{K^L-1} W_{j,i} \sum_{i=0}^{K^L-1} M_{j,k} = (WMe)_i \ .$$

For binary strings (K=2), this result was proved in Koehler (1992). For the nonbinary case, consider the binary equivalent of the strings, and further note that the length of this binary-converted string is K^L, since $K = 2^v$ for some integer v.

The expression for the spectrum of M* is derived from Lemma 1.2, by first obtaining the row sums of M. We first prove a few identities useful for this derivation.

7.5.2 Some Useful Identities

Below we give several identities that will be used throughout the remainder of this paper.

<u>Proposition 4:</u>

1. The total number of strings of length L, having exactly i zeros is

$$(K-1)^{L-1} \frac{L!}{i!(L-1)!} = (K-1)^{L-1} \binom{L}{i}$$

2. The number of strings of length L having exactly g non-zeros in the first to the p^th position (counting positions from the rightmost end of a string) is

$$\binom{p}{p-g} (K-1)^{g} K^{L-p} .$$

<u>Proof:</u> If a string has i zeros, then it must have (L-i) non-zero digits. Now (K-1) non-zero symbols may be placed in the these (L-i) places in $(K-1)^{L-i}$ ways. Also, the zeros may occur in i places out of L in L!/[i! (L-i)!] ways. Hence the first result.

The second result follows from the first. Considering the first p positions only, g non-zeros imply (p-g) zeros, and thus the number of strings with (p-g) zeros is

$$\binom{p}{p-g} (K-1)^{g} .$$

Since the remaining (L-p) positions may be occupied by any of K digits, we get the desired expression.

The following identies form the sub-expressions in the row sums calculation.

<u>Proposition 5:</u>

1. $\sum_{t=0}^{K^{L}-1} \left(\frac{\eta}{K-1}\right)^{|t|} = (1-\mu)^{-L} .$

2. $$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(t))} = K^{L-p}(1-\mu)^{-p} .$$

3. $$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=p+1}^{L}\delta(d_i(t))} = K^{p}(1-\mu)^{p-L} .$$

4. $$\sum_{t=0}^{K^L-1} \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{|s|-\Delta_{s,t,p}} = \left(\frac{\eta}{K-1}\right)^{|s|} \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{-\sum_{i=1}^{p}\delta(d_i(s))} K^{L-p} \quad (1-\mu)^{-p} .$$

5. $$\sum_{t=0}^{K^L-1} \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{|t|+\Delta_{s,t,p}} = \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{-\sum_{i=1}^{p}\delta(d_i(s))} K^{p} \quad (1-\mu)^{p-L} .$$

Proof: We first prove 1. We have,

$$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{|t|} = n_0\left(\frac{\eta}{K-1}\right)^{0} + n_1\left(\frac{\eta}{K-1}\right)^{1} + \ldots + n_L\left(\frac{\eta}{K-1}\right)^{L}$$

where n_i = number of strings with i non-zereo digits. So,

$$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{|t|} = m_L\left(\frac{\eta}{K-1}\right)^{0} + m_{L-1}\left(\frac{\eta}{K-1}\right)^{1} + \ldots + m_0\left(\frac{\eta}{K-1}\right)^{L}$$

$$= \sum_{i=0}^{L} \left(\frac{\eta}{K-1}\right)^{i} m_{L-i}$$

where m_i = number of strings with i zeros. Using Proposition 4.1 gives

$$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{|t|} = \sum_{i=0}^{L} \left(\frac{\eta}{K-1}\right)^{i} (K-1)^{i} \frac{L!}{(L-i)!\ i!} .$$

$$= \sum_{i=0}^{L} \left(\frac{\mu}{1-\mu}\right)^{i} \frac{L!}{(L-i)!\ i!}$$

$$= (1-\mu)^{-L} \sum_{i=0}^{L} \mu^{i}(1-\mu)^{L-i} \binom{L}{i}$$

$$= (1-\mu)^{-L} .$$

The last step results from noting the binomial term sums to unity. ■

Now consider the second expression. We have,

$$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(t))} = \left(\frac{\eta}{K-1}\right)^0 t_p^0 + \ldots + \left(\frac{\eta}{K-1}\right)^p t_p^p$$

where t_p^i = number of strings t having i non-zero digits from the first through the p^th position. Then, using Proposition 4.2,

$$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(t))} = \sum_{g=0}^{p} \binom{p}{g} (K-1)^g K^{L-p} \left(\frac{\eta}{K-1}\right)^g$$

$$= \sum_{g=0}^{p} \binom{p}{g} K^{L-p} \left(\frac{\mu}{1-\mu}\right)^g$$

$$= K^{L-p}(1-\mu)^{-p} \sum_{g=0}^{p} \binom{p}{g} \mu^g (1-\mu)^{p-g} .$$

Since the last binomial term sums to unity, we have the desired result. ■

Relation 5.3 is obtained similarly. We have

$$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=p+1}^{L}\delta(d_i(t))} = \sum_{h=0}^{L-p} t_{p,L}^h \left(\frac{\eta}{K-1}\right)^h$$

where $t_{p,L}^h$ gives the number of strings t with h non-zeros from the (p+1)^th to the L^th position. Using Proposition 4.2,

$$\sum_{t=0}^{K^L-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=p+1}^{L}\delta(d_i(t))} = \sum_{h=0}^{L-p} (K-1)^h \binom{L-p}{h} K^p \left(\frac{\eta}{K-1}\right)^h$$

$$= K^p \sum_{h=0}^{L-p} \binom{L-p}{h} \left(\frac{\mu}{1-\mu}\right)^h$$

$$= K^{p}(1-\mu)^{-(L-p)} \sum_{h=0}^{L-p} \binom{L-p}{h} \mu^{h}(1-\mu)^{(L-p)-h}$$

$$= K^{p}(1-\mu)^{p-L} \; . \qquad \blacksquare$$

We now consider the expression in 5.4. We know

$$\sum_{t=0}^{K^{L}-1} \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{|s|-\Delta_{s,t,p}} = \left(\frac{\eta}{K-1}\right)^{|s|} \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{-\sum_{i=1}^{p} \delta(d_i(s))} \sum_{t=0}^{K^{L}-1} \left(\frac{\eta}{K-1}\right)^{-\sum_{i=1}^{p} \delta(d_i(t))}$$

$$= \left(\frac{\eta}{K-1}\right)^{|s|} \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{-\sum_{i=1}^{p} \delta(d_i(s))} K^{L-p}(i-\mu)^{-p}$$

using Proposition 4.2.

Similarly, considering the last expression, we have

$$\sum_{t=0}^{K^{L}-1} \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{|t|+\Delta_{s,t,p}} = \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p} \delta(d_i(s))} \sum_{t=0}^{K^{L}-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=p+1}^{L} \delta(d_i(t))}$$

$$= \sum_{p=1}^{L-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p} \delta(d_i(s))} K^{p}(i-\mu)^{p-L} \qquad \blacksquare$$

Another set of useful identities relate to Walsh matrix terms in the nonbinary case.

<u>Proposition 6:</u>

1. $\sum_{x=0}^{K^{L}-1} \left(\prod_{i=0}^{L} W_{d_i(r),d_i(x)}\right) = K^{L}$ if $|r|=0$, 0 otherwise.

2. $\sum_{x=0}^{K^{p}-1} W_{r,x} = K^{p}$ if $\sum_{i=L-p+1}^{L} \delta(d_i(r)) = 0$, 0 otherwise.

<u>Proof:</u> Consider the first expression. Note that r and x are strings of length L.

Consider the digits of these strings as $r_1 \dots r_L$, and $x_1 \dots x_L$. Then in the expression

$$\sum_{x=0}^{K^{L}-1} \left(\prod_{i=0}^{L} W_{d_i(r), d_i(x)}\right)$$

there are K^L sums. Group these sums in groups of K, such that x_1 varies from 0 to (K-1) within one group, and all other x_i's remain fixed. There will be K^{L-1} such groups, each group sum is of the form

$$W_{r_L,x_L} \cdots W_{r_2,x_2}(W_{r_1,0} + W_{r_1,1} + \cdots + W_{r_1,K-1})$$

$$= K\, W_{r_L,x_L} \cdots W_{r_2,x_2} \text{ if } r_1 = 0,$$

and 0 if $r_1 \neq 0$.

If $r_1 = 0$, then consider the K^{L-2} groups where x_2 varies from 0 to (K-1). That is, now consider

$$K\, W_{r_L,x_L} \cdots W_{r_3,x_3}(W_{r_2,0} + W_{r_2,1} + \cdots + W_{r_2,K-1})$$

$$= \begin{cases} 0 \text{ if } r_2 \neq 0 \\ K^2\, W_{r_L,x_L} \cdots W_{r_3,x_3} \text{ if } r_1 = r_2 = 0 \end{cases}$$

Continuing in this manner, we finally get

If $r_1 = r_2 = \ldots = r_{L-1} = 0$, then consider the remaining K groups where x_L varies from 0 to (K-1). That is,

$$K^{L-1}(W_{r_L,0} + W_{r_L,1} + \cdots + W_{r_L,K-1})$$

$$= \begin{cases} 0 \text{ if } r_L \neq 0 \\ K^L \text{ if } r_L = r_{L-1} = \ldots = r_1 = 0 \end{cases}$$, which proves the first result. ■

Now consider the second expression:

$$\sum_{x=0}^{K^p-1} W_{r,x} = \sum_{x=0}^{K^p-1} \left(\prod_{i=0}^{L} W_{d_i(r), d_{L-i+1}(x)}\right)$$

$$= \sum_{x=0}^{K^p-1} \left(\prod_{i=0}^{L} W_{d_i(r), d_i(\text{rev}(x))}\right)$$

Since the lower (L-p) digits of rev(x) are 0s, and so the Walsh terms

$$W_{d_i(r), d_i(\text{rev}(x))} = 1$$

for these digits. Thus, the expression is

$$\sum_{x=0}^{K^p-1} W_{r,x} = \sum_{x=0}^{K^p-1} \left(\prod_{i=L-p+1}^{L} W_{d_i(r), d_i(\text{rev}(x))}\right)$$

$$= \sum_{x=0}^{K^p-1} \left(W_{r_L, x_1} W_{r_{L-1}, x_2} \ldots W_{r_{L-p+1}, x_p}\right)$$

By using Proposition 6.1, we get

$$= \begin{cases} K^p \text{ if } r_{L-p+1} = \ldots = r_{L-1} = r_L = 0 \\ 0 \text{ , otherwise.} \end{cases}$$

which gives the desired expression. ■

The following identities are necessary for obtaining the spectrum (in Section 4.5.4) from the row sums expression (derived in the next section):

Proposition 7:

1. $$\sum_{d_i(s)=0}^{K-1} W_{d_i(s),d_{L-i+1}(r)}\left(\frac{\eta}{K-1}\right)^{\delta(d_L(s))} = \begin{cases} 1-\frac{\eta}{K-1} & \text{if } d_{L-i+1}(r)\neq 0 \\ 1+\eta & \text{if } d_{L-i+1}=0 \end{cases}$$

2. $$\sum_{x=0}^{K^p-1} W_{r,x}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(x))} = \left(1-\frac{\eta}{K-1}\right)^{\sum_{i=L-p}^{L}\delta(d_i(r))}(1+\eta)^{p-\sum_{i=L-p}^{L}\delta(d_i(r))}$$

3. $$\sum_{y=0}^{K^{L-p}-1} W_{r,yK^p} = \begin{cases} K^{L-p}, & \text{if } \sum_{i=1}^{L-p}\delta(d_i(r))=0 \\ 0, & \text{otherwise} \end{cases}$$

4. $$\sum_{y=0}^{K^{L-p}-1} W_{r,yK^p}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p}\delta(d_i(y))} = \left(1-\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p}\delta(d_i(r))}(1+\eta)^{L-p-\sum_{i=1}^{L-p}\delta(d_i(r))}.$$

Proof: First consider 7.1. We have,

$$\sum_{d_i(s)=0}^{K-1} W_{d_i(s),d_{L-i+1}(r)}\left(\frac{\eta}{K-1}\right)^{\delta(d_i(s))}$$

$$= W_{0,d_{L-i+1}(r)}\left(\frac{\eta}{K-1}\right)^0 + \left(\frac{\eta}{K-1}\right)[W_{1,d_{L-i+1}(r)}+\dots+W_{K-1,d_{L-i+1}(r)}]$$

$$= \begin{cases} 1+\frac{\eta}{K-1}(-1) & \text{if } d_{L-i+1}(r)\neq 0 \\ 1+\frac{\eta}{K-1}(K-1) & \text{if } d_{L-i+1}=0 \end{cases}$$

The above expression results from noting that the Walsh coefficients are terms from the

KxK Walsh matrix, and that the row sums of the Walsh matrix, leaving out the first column values, equal K. ■

Now consider Proposition 7.2. We have

$$\sum_{x=0}^{K^p-1} W_{r,x} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(x))}$$

$$= \sum_{x=0}^{K^p-1} \left(\frac{\eta}{K-1}\right)^{\delta(d_1(x))} \left(\frac{\eta}{K-1}\right)^{\delta(d_2(x))} \cdots \left(\frac{\eta}{K-1}\right)^{\delta(d_p(x))}$$

$$[W_{d_1(x),d_L(r)} W_{d_2(x),d_{L-1}(r)} \cdots W_{d_L(x)d_1(r)}]$$

$$= \sum_{x=0}^{K^p-1} \Big\{ [(\frac{\eta}{K-1})^{\delta(d_1(x))} W_{d_1(x),d_1(rev(r))}] [(\frac{\eta}{K-1})^{\delta(d_2(x))} W_{d_2(x),d_2(rev(r))}] \cdots$$

$$\cdots [(\frac{\eta}{K-1})^{\delta(d_p(x))} W_{d_p(x),d_p(rev(r))}] [W_{d_{p+1}(x),d_{p+1}(rev(r))} \cdots W_{d_L(x),d_L(rev(r))}]$$

However, the last bracketed expression equals 1, since the (p+1)th to Lth digits of x are 0 (since x = 0 to K^p-1). Then, using Proposition 7.1, our expression becomes

$$= (1+\frac{\eta}{K-1})^{\sum_{i=1}^{p}\delta(d_i(rev(r)))} (1+\eta)^{L-\sum_{i=1}^{p}\delta(d_i(rev(r)))}$$

$$= (1+\frac{\eta}{K-1})^{\sum_{i=L-p}^{L}\delta(d_i(r))} (1+\eta)^{p-\sum_{i=L-p}^{L}\delta(d_i(r))} . \qquad ■$$

Next, considering 7.3, note that in yK^p the lower p digits of y are 0. We thus have

$$\sum_{y=0}^{K^{L-p}-1} W_{r,yK^p} = \sum_{y=0}^{K^{L-p}-1} \left(\prod_{i=1}^{L} W_{d_i(r),d_{L-i+1}(y.K^p)}\right)$$

As y varies from 0 to (K^{L-p}-1), (yK^p) varies from K^p to (K^L-1). That is, the lower p digits of (yK^p) are zero. Hence, writing $yK^p = q$, the above expression becomes

$$\sum_{y=0}^{K^{L-p}-1} W_{r,yK^p} = \sum_{q=K^p}^{K^L-1} \left(\prod_{i=1}^{L} W_{d_i(r),d_i(rev(q))}\right)$$

Note that rev(q) has its upper p digits equal to 0. So, we need consider only the lower (L-p) digits of r. Letting r_1 be the string r with upper p digits 0, we get

$$\sum_{y=0}^{K^{L-p}-1} W_{r,yK^p} = \sum_{q=K^p}^{K^L-1} \left(\prod_{i=1}^{L-p} W_{d_i(r_1),d_i(rev(q))}\right)$$

$$= \sum_{q=0}^{K^{L-p}-1} \left(\prod_{i=1}^{L-p} W_{d_i(r_1),d_i(q)}\right)$$

Using Proposition 6.1, we then get

$$\sum_{y=0}^{K^{L-p}-1} W_{r,yK^p} = \begin{cases} K^{L-p}, \text{ if all digits of } r_1 \text{ are } 0 \\ 0, \text{otherwise} \end{cases} \qquad \blacksquare$$

Finally, consider Proposition 7.4. The left hand side expression is

$$\sum_{y=0}^{K^{L-p}-1} W_{r,yK^p} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p} \delta(d_i(y))} = \sum_{y=0}^{K^{L-p}-1} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p} \delta(d_i(r))} \left[\prod_{i=1}^{L} W_{d_i(r),d_{L+i-1}(yK^p)}\right]$$

$$= \sum_{y=0}^{K^{L-p}-1} \left(\frac{\eta}{K-1}\right)^{\delta(d_1(y))} \left(\frac{\eta}{K-1}\right)^{\delta(d_2(y))} \cdots \left(\frac{\eta}{K-1}\right)^{\delta(d_{L-p}(y))}$$

$$[W_{d_1(r), d_1(rev(yK^p))} \cdots W_{d_{L-p}(r), d_{L-p}(rev(yK^p))}]$$

$$[W_{d_{L-p+1}(r), d_{L-p+1}(rev(yK^p))} \cdots W_{d_L(r), d_L(rev(yK^p))}]$$

The last bracketed expression equals 1, since d_{L-p+1} $(rev(yK^p))$ to $d_L(rev(yK^p))$ constitute the upper p digits of $rev(yK^p)$, that is, the lower p digits of (yK^p), which are all 0s. Hence the expression becomes

$$\sum_{y=0}^{K^{L-p}-1} W_{r, yK^p} \left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p} \delta(d_i(y))} = [\sum_{d_1(y)=0}^{K-1} \left(\frac{\eta}{K-1}\right)^{\delta(d_1(y))} W_{d_1(yK^p), d_1(rev(r))}] \cdots$$

$$\cdots [\sum_{d_{L-p}(y)=0}^{K-1} \left(\frac{\eta}{K-1}\right)^{\delta(d_{L-p}(y))} W_{d_{L-p}(yK^p), d_{L-p}(rev(r))}].$$

Using Proposition 7.1, this equals

$$= \left(1 - \frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p} \delta(d_i(r))} (1+\eta)^{L-p-\sum_{i=1}^{L-p} \delta(d_i(r))} . \qquad \blacksquare$$

7.5.3 Row Sums of M

The elements of M may be written as

$$r_{s,t}(0) = \frac{(i-\mu)^L}{2}\left[\left(\frac{\eta}{K-1}\right)^{|s|}\left(1-\chi+\frac{\chi}{L-1}\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{-\Delta_{s,t,p}}\right)\right.$$

$$\left.+\left(\frac{\eta}{K-1}\right)^{|t|}\left(1-\chi+\frac{\chi}{L-1}\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{\Delta_{s,t,p}}\right)\right].$$

For obtaining the row-sums of M, we sum the above expression from $t = 0$ to $K^L - 1$.

Now,

$$\sum_{t=0}^{K^L-1}\frac{(1-\mu)^L}{2}\left[\left(\frac{\eta}{K-1}\right)^{|s|}(1-\chi)+\left(\frac{\eta}{K-1}\right)^{|t|}(1-\chi)\right]$$

$$= K^L\frac{(1-\mu)^L}{2}\left(\frac{\eta}{K-1}\right)^{|s|}(1-\chi)+\frac{(1-\mu)^L}{2}(1-\chi)\sum_{t=0}^{K^L-1}\left(\frac{\eta}{K-1}\right)^{|t|}$$

$$= K^L\frac{(1-\mu)^L}{2}\left(\frac{\eta}{K-1}\right)^{|s|}(1-\chi)+\frac{(1-\chi)}{2}$$

using Proposition 5.1.

Also, combining Propositions 5.4 and 5.5, we get

$$\frac{(1-\mu)^L}{2}\frac{\chi}{L-1}\left[\sum_{t=0}^{K^L-1}\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|s|-\Delta_{s,t,p}}+\sum_{t=0}^{K^L-1}\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|t|+\Delta_{s,t,p}}\right.$$

$$= \frac{(1-\mu)^L}{2}\frac{\chi}{L-1}\left[\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{p}\delta(d_i(s))}K^{L-p}(1-\mu)^{-p}+\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))}K^{p}(1-\mu)^{p-L}\right]$$

$$= \frac{\chi}{2(L-1)}\left[\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{p}\delta(d_i(s))}(K-K\mu)^{L-p}+\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))}(K-K\mu)^{p}\right]$$

$$= \frac{\chi}{2(L-1)}\left[\sum_{h=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{L-h}\delta(d_i(s))}(K-K\mu)^{h}+\sum_{p=1}^{L-1}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))}(K-K\mu)^{p}\right]$$

$$= \frac{\chi}{2(L-1)}\left(\sum_{p=1}^{L-1}(K-K\mu)^{p}\left[\left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{L-p}\delta(d_i(s))}+\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))}\right]\right)$$

$$= \frac{\chi}{2(L-1)}\sum_{p=1}^{L-1}(K-K\mu)^{p}\,G(p,s)$$

where

$$G(p,s) = \left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{L-p}\delta(d_i(s))}+\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))}.$$

Then combining the two expressions obtained above, we get the expression for the row sums:

$$\sum_{t=0}^{K^L-1} r_{s,t}(0) = \frac{(1-\chi)}{2}+K^{L}\frac{(1-\mu)^L}{2}\left(\frac{\eta}{K-1}\right)^{|s|}(1-\chi)+\frac{\chi}{2(L-1)}\sum_{p=1}^{L-1}(K-K\mu)^{p}\,G(p.s).$$

7.5.4 Expression for the Spectrum

The following terms comprise prominent sub-expressions in the derivation of the spectrum:

Proposition 8:

1. $$\sum_{s=0}^{K^L-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L}\delta(d_i(s))} = \left(1-\frac{\eta}{K-1}\right)^{|r|}(1+\eta)^{L-|r|}$$

2. $$\sum_{s=0}^{K^L-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))} = \begin{cases} K^{L-p}\left(1-\frac{\eta}{K-1}\right)^{|r|}(1+\eta)^{p-|r|} & \text{if } \sum_{i=1}^{L-p}\delta(d_i(r))=0 \\ 0 \text{ otherwise.} \end{cases}$$

3. $$\sum_{s=0}^{K^L-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{p}\delta(d_i(s))} = \begin{cases} K^{p}\left(1-\frac{\eta}{K-1}\right)^{|r|}(1+\eta)^{L-p-|r|} & \text{if } \sum_{i=L-p+1}^{L}\delta(d_i(r))=0 \\ 0 \text{ otherwise.} \end{cases}$$

Proof:

First consider 8.1 The left hand side expession is

$$\sum_{s=0}^{K^L-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L}\delta(d_i(s))} = \sum_{s=0}^{K^L-1}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L}\delta(d_i(s))}\left[\prod_{i=1}^{L} W_{d_i(s),d_{L-i+1}(r)}\right]$$

$$= \left[\sum_{d_1(s)=0}^{K-1} W_{d_1(s),d_L(r)}\left(\frac{\eta}{K-1}\right)^{\delta(d_1(s))}\right]\left[\sum_{d_2(s)=0}^{K-1} W_{d_2(s),d_L(r)}\left(\frac{\eta}{K-1}\right)^{\delta(d_2(s))}\right]\ldots$$

$$\ldots\left[\sum_{d_L(s)=0}^{K-1} W_{d_L(s),d_L(r)}\left(\frac{\eta}{K-1}\right)^{\delta(d_L(s))}\right]$$

Using Proposition 4.1, this expression becomes

$$= \left(1-\frac{\eta}{k-1}\right)^{|r|}(1+\eta)^{L-|r|}. \qquad \blacksquare$$

Now considering 8.2, the left hand side expression may be written as

$$\sum_{s=0}^{K^L-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))} = \sum_{y=0}^{K^{L-p}-1}\sum_{x=0}^{K^p-1} W_{r,x+yK^p}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))}$$

$$= \sum_{y=0}^{K^{L-p}-1} W_{r,yK^p}\sum_{x=0}^{K^p-1} W_{r,x}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{p}\delta(d_i(s))}$$

Using Proposition 4.2 for the second summation above we get,

$$= \sum_{y=0}^{K^{L-p}-1} W_{r,yK^p}\left[\left(1-\frac{\eta}{K-1}\right)^{\sum_{i=L-p}^{L}\delta(d_i(r))}(1+\eta)^{p-\sum_{i=L-p}^{L}\delta(d_i(r))}\right]$$

Using the result of Proposition 4.4, this becomes

$$= \begin{cases} K^{L-p}\left(1-\frac{\eta}{K-1}\right)^{\sum_{i=L-p}^{L}\delta(d_i(r))}(1+\eta)^{p-\sum_{i=L-p}^{L}\delta(d_i(r))} \\ 0, \text{ otherwise .}\end{cases}$$

But if

$$\sum_{i=1}^{L-p}\delta(d_i(r)) = 0,$$

then $\sum_{i=L-p}^{L}\delta(d_i(r)) = \sum_{i=1}^{L}\delta(d_i(r)) = |r|$.

Thus, we get

$$= \begin{cases} K^{L-p}\left(1-\frac{\eta}{K-1}\right)^{|r|}(1+\eta)^{p-|r|} \text{ if } \sum_{i=1}^{L-p}\delta(d_i(r))=0 \\ 0, \text{ otherwise.}\end{cases} \qquad \blacksquare$$

Finally, consider 8.3. The left hand side expression is

$$\sum_{s=0}^{K^{L}-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{p}\delta(d_i(s))} = \sum_{s=0}^{K^{L}-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{\sum_{i=p+1}^{L}\delta(d_i(s))}$$

$$= \sum_{y=0}^{K^{L-p}-1}\ \sum_{x=0}^{K^{p}-1} W_{r,x+yK^{p}}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p}\delta(d_i(y))}$$

$$= \sum_{x=0}^{K^{p}-1} W_{r,x}\left[\sum_{y=0}^{K^{L-p}-1} W_{r,yK^{p}}\left(\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p}\delta(d_i(y))}\right]$$

Using Proposition 4.4, this becomes

$$\sum_{s=0}^{K^{L}-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{p}\delta(d_i(s))} = \sum_{x=0}^{K^{p}-1} W_{r,x}\left[\left(1-\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p}\delta(d_i(r))}(1+\eta)^{L-p-\sum_{i=1}^{L-p}\delta(d_i(r)}\right]$$

Now using the identity of Proposition 6.2, we have,

$$\sum_{s=0}^{K^{L}-1} W_{r,s}\left(\frac{\eta}{K-1}\right)^{|s|-\sum_{i=1}^{p}\delta(d_i(s))} = \begin{cases} K^{p}\left(1-\frac{\eta}{K-1}\right)^{\sum_{i=1}^{L-p}\delta d_i(r))}(1+\eta)^{L-p-\sum_{i=1}^{L-p}\delta d_i(r))} & \text{if } \sum_{i=L-p+1}^{L}\delta d_i(r))=0 \\ 0, \text{ otherwise} . \end{cases}$$

But if

$$\sum_{i=1}^{L-p+1}\delta(d_i(r)) = 0,$$

we get $\sum_{i=1}^{L-p}\delta(d_i(r)) = \sum_{i=1}^{L}\delta(d_i(r)) = |r|$.

Hence, the result. ■

The expression for the spectrum may now be obtained using the above results:

If S_s denotes the s^{th} row sum of M, then from Eq. 5.3.4, we have

$$\sum_s W_{r,s} S_s = \frac{(1-\chi)}{2} \sum_s W_{r,s} + K^L \left(\frac{1-\mu^L}{2}\right)(1-\chi) \sum_s W_{r,s} \left(\frac{\eta}{K-1}\right)^{|s|} + \frac{\chi}{2(L-1)} \sum_{p=1}^{L-1} (K-K\mu)^p \sum_s W_{r,s} G(p,s).$$

Consider the term

$$\sum_{p=1}^{L-1} (K-K\mu)^p \sum_s W_{r,s} G(p,s)$$

$$= \sum_{p=1}^{L-1} (K-K\mu)^p \Big[K^{L-p} \left(1-\frac{\eta}{K-1}\right)^{|r|} (1+\eta)^{p-|r|} \operatorname{del}\Big(\sum_{i=1}^{L-p} \delta(d_i(r))\Big)$$

$$+ K^{L-p} \left(1-\frac{\eta}{K-1}\right)^{|r|} (1+\eta)^{p-|r|} \operatorname{del}\Big(\sum_{i=L-p+1}^{L} \delta(d_i(r))\Big)\Big]$$

$$= \sum_{p=1}^{L-1} K^p (1-\mu)^p k^{L-p} \left(1-\frac{\eta}{K-1}\right)^{|r|} (1+\eta)^{p-|r|} \Big[\operatorname{del}\Big(\sum_{i=1}^{p} \delta(d_i(r))\Big) + \operatorname{del}\Big(\sum_{i=p+1}^{L} \delta(d_i(r))\Big)\Big]$$

$$= K^L (1-K\mu)^{|r|} \sum_{p=1}^{L-1} \Big[\operatorname{del}\Big(\sum_{i=1}^{p} \delta(d_i(r))\Big) + \operatorname{del}\Big(\sum_{i=p+1}^{L} \delta(d_i(r))\Big)\Big]$$

When $r = 0$, this expression equals $K^L\ 2\ (L - 1)$.

When $r > 0$, the last summation equals $[\ n_{\text{right}}\ (1+0)\ +\ \text{wid}(r)\ +\ n_{\text{left}}\ (0 + 1)\ ]$, where

n_{right} = number of trailing 0's in r,

n_{left} = number leading 0s, leaving out the left-most digit, in r,

and, wid (r) is the same as the defining length of r.

Thus, we have

$$\sum_{p=1}^{L-1} (K-K\mu)^p \sum_s W_{r,s} G(p,s) = K^L (1-K\mu)^{|r|} [L-1-\text{wid}(r)].$$

So, when $r = 0$,

$$\sum_s W_{r,s} S_s = \frac{(1-\chi)}{2} K^L + K^L \frac{(1-\mu)^L}{2} (1-\chi)(1+\eta)^L + \frac{\chi}{2(L-1)} K^L 2(L-1)$$

Then by Lemma 1.2, the eigenvalue of M* corresponding to $r = 0$, is obtained by dividing the above expression by K^L, which gives

$$\frac{1-\chi}{2}+\frac{1-\chi}{2}+\chi = 1.$$

When $r > 0$,

$$\sum_s W_{r,s}S_s = K^L\left(\frac{1-\mu^L}{2}\right)(1-\chi)\left(1-\frac{\eta}{K-1}\right)^{|r|}(1+\eta)^{L-|r|}+\frac{\chi}{2(L-1)}K^L(1-K\mu)^{|r|}(L-1-\mathrm{wid}(r))$$

$$= K^L\frac{(1-\chi)}{2}\left(1-\frac{K\mu}{K-1}\right)^{|r|}+\frac{\chi}{2(L-1)}K^L(1-K\mu)^{|r|}(L-1-\mathrm{wid}(r))$$

Again, by Lemma 1.2, the eigenvalues of M* can be obtained by dividing this expression by K^L giving

$$\frac{(1-\chi)}{2}\left(1-\frac{K\mu}{K-1}\right)^{|r|}+\frac{\chi}{2(L-1)}(1-K\mu)^{|r|}(L-1-\mathrm{wid}(r))$$

which is decreasing in $|r|$ and in wid (r) when $\mu < 1/K$.

The second largest eigenvalue corresponds to $r = 1$:

$$\frac{(1-\chi)}{2}\left(1-\frac{K\mu}{K-1}\right)+\frac{\chi}{2(L-1)}(1-K\mu)(L-1)$$

This simplifies to the form

$$\frac{1}{2}\left[1-\frac{K}{K-1}\mu-\chi K\mu\left(\frac{K-2}{K-1}\right)\right]$$

which is $<\frac{1}{2}$ for $\mu < 1/K$.

Thus, second largest eigenvalue of M^* is less than 1/2, for $\mu < 1/K$. . By Proposition 3, this implies that, for nonbinary, higher cardinality representations, the fixed points of the recombination operator $\bar{M}$ are stable irrespective of the crossover rate when $\mu < 1/K$.

CHAPTER 8
SUMMARY AND FUTURE RESEARCH

8.1 Summary

The complexity of the scheduling problem requires the application of sophisticated techniques for handling real world applications. The limitations of traditional OR approaches have been discussed in Section 2.1. In recent years, a variety of AI and knowledge based techniques have been proposed for modelling the production environment, with improved productivity being reported in a number of factory floor applications. The potential advantages of AI for production scheduling has been outlined in Section 2.3, and Section 2.4 reviews various reported knowledge based approaches to the scheduling problem.

With the utilization of AI schemes, however, there has also come about a realization of their limitations. Foremost amongst these is the difficulty in obtaining the requisite knowledge for the systems to be able to effectively schedule various production tasks. The knowledge acquisition bottleneck, widely recognized as a major deterrent to the development of knowledge based solution procedures in other application domains, is all the more severe in the production scheduling environment due to (1) the non existence of true experts in many manufacturing environments, and (2) complexity of the scheduling problem being beyond the cognitive abilities of human schedulers, who are thus led to using myopic strategies and incomplete information.

Consequently, the development of automated means for acquiring the required scheduling knowledge has become a primary research concern. The need for learning mechanisms for knowledge based scheduling has been elaborated in Section 3.1, and the literature in machine learning applications to production scheduling reviewed in Section 3.2. A discussion of related issues of importance to this relatively new field of research in machine learning applications to scheduling is given in Section 3.3.

The research presented here focussed on the development of an automated knowledge acquisition facility within a factory scheduling environment. An adaptive decision support framework for the scheduling of tasks in typically decentralized factory environment, through the use of multiple intelligent objects, has been implemented. Here, genetic algorithm based learning is utilized. A knowledge representation scheme allowing both an effective modelling of the manufacturing environment and manipulation through genetic search operators has been designed. Genetic learning facilities, including a special purpose crossover operator, voting based inference, and credit assignment policies have been developed. The implemented system is detailed in Chapter 5.

A number of simulation experiments were conducted to demonstrate the feasibility of the proposed scheme, and determine the sensitivity of the developed system to various parameters. The experiments provided valuable insights into the behavior of the system, and revealed certain drawbacks requiring enhancements to the existing facilities. A couple of such modifications were implemented, and the results presented in Chapter 6 show significant improvements in scheduling performance with learning. A number of additional features remain to be implemented, and these with other future research issues are discussed in Section 8.2.1.

Most theoretical analyses of the behavior of genetic algorithms consider binary representations, and relatively few studies have been done on genetic search over high level representations. A number of applications, including that presented in this research, use high level representation schemes. A second part of this research thus undertook a theoretical study of nonbinary genetic algorithms. A generalization of the mathematical formalization of binary- GA search behavior, as provided by Vose and Liepins (1991), to GAs using higher cardinality alphabets, has been obtained. Related future research issues are discussed in Section 8.2.2.

8.2 Future Research

8.2.1 Genetic Learning of Scheduling Strategies

Experiments with the developed system, presented in Chapter 6, demonstrate the feasibility of the designed approach in effecting adaptive scheduling behavior in response to a given manufacturing environment and stated performance objectives. Insights into the system's behavior point out avenues for further improvement.

A mechanism is necessary for overcoming the degeneracy problem explained in Section 6.4. The utility of a more sophisticated voting mechanism, possibly taking into account the variances of the rules/conjuncts' fitness values, (as described in Greffenstette, 1987) needs to be examined. The employed voting scheme using actual and virtual vote distributions incorporates an inherent bias against high performing conjuncts which do not, however, form part of a consistent rule. Such a conjunct, in the presence of any other consistent rule (even though these may not perform as well as the stated conjunct), will never take part in the primary job selection decision since they do not assign actual votes. Any payoff received by the conjunct is purely on chance, when the chosen job happens to be that which this conjunct assigns virtual votes to. A potentially good conjunct can, in this manner, fail to accumulate strength and finally be forced out of the knowledge base. A flat voting scheme, using a single vote distribution, with possibly higher weightage to consistent rules, may help overcome this problem.

A single crossover operator has been used in the simulations so far. A variety of crossover mechanisms may be designed, and these together with other selection schemes need to be considered with the developed representation. A facility whereby rules and conjuncts carry a proportion of their fitnesses to the next generation may also prove beneficial. The experiments indicated that a higher than usual mutation rate of 0.2 yielded good results. High mutation, as noted in Chapter 6, can, however, cause instability in the knowledge base. A variable mutation rate mechanism, based on how well a knowledge base is currently performing, can effect an adequate tradeoff.

The incorporation of attributes and atoms permits flexibility in the representable knowledge. Atoms have proved useful in the formulation of condition-action rules. More effective schemes for learning rules comprising of high performing atoms need to be investigated.

In a dynamic environment like the one considered here, different sets of rules may perform well in dissimilar situations, and the knowledge base may not stabilize or converge. A long term memory, retaining the best performing rules over the period of the simulation, has several potential benefits in this situation. When changes in the environment cause the current knowledge base to perform poorly, the knowledge base may be seeded with rules from the long-term memory, thereby allowing previously well-performing rules enter into the decision making process. When tagged with the state of the environment, rules in long-term memory also provide a set of rules that perform well in different situations. It may also prove easier to use attributes and atoms to map the state of the environment at different points in time and associate these with the high fitness rules in long-term memory, than to attempt to evolve rules with atoms indicating the current state.

Finally, as explained in Section 5.2, multiple evaluation points are envisioned for scheduling in a complex environment. With evaluation and credit assignment being crucial for effective GA performance, the establishment of intermediate evaluation points, and the formulation of appropriate evaluation criteria is an important issue for research. Modelling of fitness functions that adequately reflect scheduling criterion is also in itself a topic of significance to the use of GAs for production scheduling.

8.2.2 Analysis of Nonbinary GAs

The theoretical model proposed by Vose and Liepins (1991) provides a detailed characterization of the search behavior of binary coded GAs. Formalizing the genetic operators of crossover and mutation as a dispersion operator, and fitness-proportionate selection as a focussing operator, they provide a precise description of the punctuated equilibria that is typically noticed in genetic search. A crucial conjecture relating to the asymptotic stability of the fixed points of the search operators has since been proved in

Koehler (1992). In this research, generalized results have been derived for GAs employing higher cardinality representations, including Koehler's (1992) result.

The results obtained here allow a direct comparison of binary versus higher cardinality string encodings. Walsh functions have been invaluable in the theoretical analyses of binary GAs (Goldberg, 1989b, 1989c). This research provides a generalization of the Walsh matrix terms when considering higher cardinality representations. This, together with other identities derived in the process of the analysis should prove useful in the study of nonbinary GAs. Analysis of modified crossover and mutation operators, and of deceptiveness, in nonbinary GAs, are important areas for future research.

The obtained higher-cardinality representation model also allows the generalization of the Markov chain model of genetic search provided in Nix and Vose (1992). Aytug and Koehler (1993) use this Markov chain model to obtain bounds on the run-time complexity of binary GAs. The results obtained in this research also allow the extension of these bounds to the nonbinary case.

REFERENCES

Alpar, P., and K.N. Srikanth (1989), "Closed-Shop Scheduling with Expert System Techniques," *Knowledge-based Systems in Manufacturing*, A. Kusiak (ed.), Taylor and Francis, New York, pp. 247-264.

Ammons, J.C., T. Govindaraj and C.M. Mitchell (1988), "A Supervisory Control Paradigm for Real-Time Control of Flexible Manufacturing Systems," *Annals of Operations Research*, 15, pp. 313-335.

Antonisse, H.J. (1989), "A New Interpretation of Schema that Overturns the Binary Encoding Constraint," *Proceedings of the Third International Conference of Genetic Algorithms*, D.Schaefer (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 86-91.

Antonisse, H.J. and K.S. Keller (1987), "Genetic Operators for High Level Representations", *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms*, MIT, J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 69-76.

Arff, S., and G. Hasle (1990), "Synthesis of Schedules Using Heuristic, Constraint-Guided Search," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C., May, pp. 111-124.

Aytug, H., G.J. Koehler and J.L Snowdon (1993), "Genetic Learning of Dynamic Scheduling within a Simulation Environment," *Computers and Operations Research*, forthcoming.

Baker, J.E. (1985), "Adaptive Selection Methods for Genetic Algorithms", *Proceedings of the First International Conference on Genetics Algorithms and their Applications*, J.J. Grefenstette (ed.), Carnegie-Mellon Univ., Pittsburgh, pp. 100-111.

Baker, J.E. (1987), "Reducing Bias and Ineffeciency in the Selection Algorithm," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms*, MIT, J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 14-21.

Baker, T. E. (1990), "Integrating AI/OR/Database Technologies for Production Planning and Scheduling," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C May, pp. 101-110.

Bel, G., E. Bensana, D. Dubois, J. Erschier, P. and Esquirol (1989), "A Knowledge-Based Approach to Industrial Job-Shop Scheduling," *Knowledge-Based Systems in Manufacturing*, A. Kusiak (ed.), Taylor & Francis, New York, pp. 207-246.

Ben-Arieh, D. (1986), "Knowledge-Based Control System for Automated Production and Assembly," *Modelling and Design of Flexible Manufacturing Systems*, A. Kusiak (ed.), Elsevier, New York, pp. 347-368.

Bensana, E., M. Correge, G. Bel, and D. Dubois (1986), "An Expert-System Approach to Industrial Job-Shop Scheduling," *Proceedings of 1986 IEEE International Conference on Robotics and Automation*, San Francisco, April 7-10, pp. 1645-1650.

Bitran, G. and T. Papageorge (1988), "Integration of Manufacturing Policy and Corporate Strategy with the Aid of Expert Systems," *Intelligent Manufacturing, Proceedings of the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, M.D. Oliff (ed.), Benjamin/Cummings, San Francisco, pp. 13-43.

Blackstone, J. H., D.T. Phillips, and G.L. Hogg (1982), "The State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations," *International Journal of Production Research*, 20, No. 1, pp. 27-45.

Blessing, J.A., and B.A. Watford (1987), "INFMSS, An Intelligent FMS Scheduling System," *World Productivity Forum and 1987 Annual International Industrial Engineering Conference Proceedings*, New York, pp. 82-88.

Booker, L.B. (1987), "Improving Search in Genetic Algorithms," *Genetic Algorithms and Simulated Annealing*, L.Davis (ed.), Pitman, London, pp. 61-78.

Booker, L.B. (1991), "Representing Attribute-Based Concepts in a Classifier System," *Foundations of Genetic Algorithms*, Gregory J.E. Rawlins (ed.), Morgan Kaufmann Pub., San Mateo, Calif, pp. 115-127.

Bruno, B., A. Elia, and P. Laface (1986), "A Rule-Based System to Schedule Production," *IEEE Computer*, 19, No. 7, pp. 32-44.

Bu-Hulaiga, M.I. and A.K. Chakravarty (1988), "An Object-Oriented Knowledge Representation for Hierarchical Real-Time Control of Flexible Manufacturing," *Int. J. Prod. Res.*, 26, No. 5, pp. 777-793.

Bullers, W.I., S.Y. Nof and A. B. Whinston (1980), "Artificial Intelligence in Manufacturing Planning Control," *AIIE Transactions*, 12, No. 4, pp. 351-363.

Buxey, G. (1989), "Production Scheduling: Practice and Theory," *European Journal of Operational Research*, 39, pp. 17-31.

Chiang, W.Y., and M.S. Fox (1990), "Protection Against Uncertainty in a Deterministic Schedule," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S. C., May, pp. 184-197.

Chiang, W-Y, M.S. Fox, and P.S. Ow (1990), "Factory Model and Test Data Descriptions: OPIS Experiments," Report No. CMU-RI-TR-90-05, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Chiodini, V. (1986), "An Expert System For Dynamic Manufacturing Rescheduling," *Symposium on Real Time Optimization in Automated Manufacturing Facilities,* National Bureau of Standards, Gaithersburg, Md., January, pp 86-91.

Chryssolouris, G., K. Wright, and J. Pierce (1988a), "A Simulator for Manufacturing Systems Based on Artificial Intelligence," *Proceedings of the Winter Annual Meeting of the American Society of Mechanical Engineers,* Chicago, pp. 1-13.

Chryssolouris, G., K. Wright, J. Pierce and W. Cobb (1988b), "Manufacturing Systems Operation: Dispatch Rules Versus Intelligent Control," *Robotics and Computer-Integrated Manufacturing,* 4, No. 3/4, pp. 531-544.

Chryssolouris, G., J.E. Pierce and W. Cobb (1989), "A Decision-Making Approach to the Operation of FMS," *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems:Operations Research Models andApplications,* K.E. Steke and R. Suri (eds.), Elsevier Science Pub., Amsterdam, pp. 355-360.

Chryssolouris, G., M. Lee and M. Domroese (1990), "The Use of Neural Networks in Determining Operational Policies for Manufacturing Systems," *Journal of Manufacturing Systems,* 10, No. 2, pp. 166-175.

Chryssolouris, G., M. Lee, J. Pierce, and M. Domroese (1990), "Use of Neural Networks for the Design of Manufacturing Systems," *Manufacturing Review,* 3, No., 3, pp. 187-194.

Chryssolouris, G., S. Graves, and K. Ulrich (1991), "Decision Making in Manufacturing Systems: Product Design Production Planning and Process Control," *Proceedings of the 1991 NSF Design and Manufacturing Systems Conference*, Austin, Tx., pp. 693-701.

Clancy, D.P. and S. Mohan (1990), "RBD-Rule Based Job Dispatching Software for Implementing Production Plans," *First International Conference on Expert Planning Systems*, Pittsburgh, June, pp. 100-103.

Cleveland, G.A. and S.F. Smith (1989), "Using Genetic Algorithms to Schedule Flow Shop Releases," *Proceedings of the Third International Conference on Genetic Algorithms*, D.Schaefer (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 160-169.

Coombs, S. and L. Davis (1987), "Genetic Algorithms and Communication Link Speed Design: Constraints and Operators," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms*, MIT, J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 257-260.

Czigler, M.H., and C.R. Whitaker (1990), "A Hybrid Algorithmic and Knowledge-Based Implementation for Workcenter-Based Production Scheduling," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C., May 1990, pp. 145-155.

Davis, L. (1985), "Job Shop Scheduling with Genetic Algorithms," *Proceedings of the First International Conference on Genetics and their Applications*, J.J. Grefenstette (ed.), Carnegie-Mellon Univ., Pittsburgh, pp.136-140.

Davis, L. and S. Coombs (1987), "Genetic Algorithms and Communication Link Speed Design: Theoretical Considerations," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms*, MIT, J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 252-256.

Davis, R. and R. Smith (1983), "Negotiation as a Metaphor for Distributed Problem Solving," *Artificial Intelligence*, 20, pp. 63-109.

De, S. (1988), "A Knowledge-based Approach to Scheduling in an FMS," *Annals of Operations Research*, 12, pp. 109-134.

De, S., and A. Lee (1990), "A Knowledge-Based System for Flexible Assembly Scheduling," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C., May, pp. 156-167.

DeJong, K. (1988), "Learning with Genetic Algorithms: An Overview," *Machine Learning*, 3, pp. 121-138.

DeJong, K. and W.M. Spears (1990), "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms," *Proceedings of the First International Conference on Parallel Problem Solving from Nature*, IEEE Society Press., Dortmund, Germany, pp. 38-47.

Donath, M., and R.J. Graves (1988), "Flexible Assembly Systems: An Approach for Near Real-time Scheduling and Routing of Multiple Products," *International Journal of Production Research*, 26, No. 12, pp. 1903-1919.

Duchessi, P. (1987), "The Conceptual Design for a Knowledge-Based System AI Applied to the Production Planning Process," *Expert Systems for Business*, D. Silverman (ed.), Addison-Wesley, Reading, Mass., pp. 131-143.

Duchessi, P., S. Belardo, and J. Seagle (1988), "Knowledge Enhancements to a Decision Support System for Vehicle Routing," *Inferfaces*, 18, No. 2, pp. 21-29.

Ebner, M.L. and T.E. Vollmann (1988), "Manufacturing Systems for the 1990's," *Intelligent Manufacturing*, M.D. Oliff (ed.), Benjamin/Cummings Publishing Company, pp. 317-335.

Erschler, J. and P. Esquirol (1986), "Decision-Aid in Job Shop Scheduling: A Knowledge Based Approach," *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, April 7-10, pp. 1651-1656.

Erschler, J. and F. Roubellat (1989), "An Approach to Solve Workshop Real Time Scheduling Problems," *Advanced Information Technologies for Industrial Material Flow Systems*, S.Y. Nof and C.L. Moodie (eds.), Springer-Verlag, Berlin, pp. 651-679.

Eshelman, L.J. (1991), "Spurious Correlations and Premature Convergence in Genetic Algorithms," *Foundations of Genetic Algorithms*, Gregory J.E. Rawlins (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 1032-112.

Eshelman, L.J., R.A. Caruna and J. D. Schaffer (1989), "Biases in the Crossover Landscape," *Proceedings of the Third International Conference on Genetic Algorithms*, J. David Schaffer (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 10-19.

Farhoodi, F. (1990), "A Knowledge-Based Approach to Dynamic Job-Shop Scheduling," *International Journal of Computer Integrated Manufacturing*, Vol. 3, No. 2, pp. 84-95.

Fiegenbaum, E., P. McCorduck and H.P. Nii (1988), *The Rise of the Expert Company: How Visionary Companies Are Using Artificial Intelligence to Achieve Higher Productivity and Profits*, Times Books, New York, pp. 55-64.

Fordyce, K., P. Norden, and G. Sullivan (1987), "Artificial Intelligence and the Management Science Practitioner: Links Between Operations Research and Expert Systems," *Interfaces*, 17, No. 4, pp. 34-40.

Fordyce, K. and G. Sullivan (1989), "Logistics Management System (LMS): Implementing the Technology of Logistics with Knowledge Based Expert Systems," *Innovative Expert System Application*, AAAI and MIT Press, Menlo Park, Calif., pp. 36-51.

Forrest, S. (1985), "Implementing Semantic Network Structures Using the Classifier System," *Proceedings of the First International Conference on Genetic Algorithms*, J.J Grefenstette (ed.), Carnegie-Mellon Univ., Pittsburgh, pp. 24-44.

Forrest, S. and G. Mayer-Kress (1991), "Genetic Algorithms, Nonlinear Dynamical Systems, and Models of International Security," *Handbook of Genetic Algorithms*, L. Davis (ed.), Ch. 13, Van Nostrand Reinhold Pub., New York, pp. 166-185.

Fox, B.R. and K.G. Kempf (1985a), "Opportunistic Scheduling for Robotic Assembly," *IEEE 1985 International Conference on Robotics and Automation*, Institute of Electrical and Electronic Engineers, St. Louis, March, pp. 880-889.

Fox, B.R. and K.G. Kempf (1985b), "Uncertainty and Opportunistic Scheduling," *Second Conference on Artificial Intelligence Applications*, IEEE Computer Society, Miami Beach, Fla., December, pp. 487-492.

Fox, B.R., and K.G. Kempf (1987), "Reasoning About Opportunistic Schedules," *IEEE*, pp. 1876-1882.

Fox, B.R. and M.B. McMahon (1991), "Genetic Operators for Sequencing Problems," *Foundations of Genetic Algorithms*, G.J.E. Rawlins (ed.), Morgan Kaufmann Pub., San Mateo,Calif., pp. 284-300.

Fox, M. S. (1990), "Constraint-Guided Scheduling--A Short History of Research at CMU," *Computers in Industry*, 14, pp. 79-88.

Fox, M. S., N. Sadeh, and C. Baykan (1989), "Constrained Heuristic Search," *Eleventh International Joint Conference on Artificial Intelligence*, Detroit, August 20-25, pp. 309-315.

Fox, M.S. and N. Sadeh (1990), "Why is Scheduling Difficult? A CSP Perspective," *Proceedings of the 9th European Conference on Artificial Intelligence*, Stockholm, Sweden, August 6-10, pp. 754-767.

Fox, M.S., and S.F. Smith (1984a), "The Role of Intelligent Reactive Processing In Production Management," *Proceedings of CAM-I's 13th Annual Meeting and Technical Conference*, Clearwater Beach, Fla., November 13-15, pp. 13-17.

Fox, M.S., and S.F. Smith (1984b), "ISIS-A Knowledge-Based System for Factory Scheduling," *Expert Systems*, 1, No. 1, pp. 25-44.

Fox, M. S. and K.P. Sycara (1990), "Overview of CORTES: A Constraint Based Approach to Production Planning, Scheduling and Control," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C., May, pp. 84-96.

Fukuda, T., S. Takeda and M. Hayashi (1986), "Distributed Expert Systems for Production Control," *1986 International Industrial Engineering Conference*, Institute of Industrial Engineers, Dallas, May, pp. 222-228.

Gere, W.S. Jr. (1962), "A Heuristic Approach to Job Shop Scheduling," Ph.D. Thesis, Graduate School of Industrial Administration, Carnegie Institute of Technology, Pittsburg.

Goldberg, D.E. (1985), "Genetic Algorithms and Rule Learning in Dynamic System Control," *Proceedings of the First International Conference on Genetics and their Applications*, J.J. Grefenstette (ed.), Carnegie-Mellon Univ., pp. 8-15.

Goldberg, D.E. (1987), "Simple Genetic Algorithms and the Minimal Deceptive Problem," *Genetic Algorithms and Simulated Annealing*, L. Davis (ed.), Pitman Pub., London, pp. 74-88.

Goldberg, D.E. (1989a), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, Mass.

Goldberg, D.E. (1989b), "Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction," *Complex Systems*, 3, pp. 129-152.

Goldberg, D.E. (1989c), "Genetic Algorithms and Walsh Functions: Part II, Deception and its Analysis," *Complex Systems*, 3, pp. 153-171.

Goldberg, D.E. (1990a), "Probability Matching, the Magnitude of Reinforcement, and Classifier System Bidding," *Machine Learning*, 5, pp. 407-425.

Goldberg, D.E. (1990b), "Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking," Tech. Report IlliGAL No. 90001, Univ. of Illinois at Urbana-Champaign.

Goldberg, D.E. (1990c), "Construction of High-Order Deceptive Functions Using Low Order Walsh Coefficients," Tech. Report IlliGAL No. 900002, Univ. of Illinois at Urbana-Champaign.

Goldberg, D.E. and K. Deb (1991), "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," *Foundations of Genetic Algorithms,* G.J.E. Rawlins (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp.69-93.

Goldberg, D.E., K. Deb and B. Korb (1990), "Messy Genetic Algorithms Revisited: Studies in Mixed Size and Scale," *Complex Systems,* 4, pp. 415-444.

Goldberg, D.E., B. Korb and K. Deb (1989), "Messy Genetic Algorithms: Motivation, Analysis, and First Results," *Complex Systems,* 3, pp. 493-530.

Goldberg, D.E. and R. Lingle (1985), "Alleles, Loci, and the Travelling Salesman Problem," *Proceedings of the First International Conference on Genetics and their Applications,* J.J. Grefenstette (ed.), Carnegie-Mellon Univ., pp. 154-159.

Grant, T.J. (1986), "Lessons for O.R. From A.I.: A Scheduling Case Study," *Journal of Operational Research Society,* 1986, 37, No. 1, pp. 41-57.

Graves, S.C. (1981), "A Review of Production Scheduling," *Operations Research,* 29, N
14, pp. 646-675.

Greene, D.P. and S.F. Smith (1987), "A Genetic System for Learning Models of Consumer Choice," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms,* MIT, J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 216-223.

Grefenstette, J.J. (1987), "Multilevel Credit Assignment in a Genetic Algorithm Learning System," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms,* MIT, J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 202-209.

Grefenstette, J.J. (1988), "Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms," *Machine Learning,* 3, pp. 225-245.

Grefenstette, J.J. (1989), "Incremental Learning of Control Strategies with Genetic Algorithms," *Proceedings of the Sixth International Workshop on Machin*e Learning, Cornell Univ., A. M. Segre (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 340-344.

Gupta, T. and B.K. Ghosh (1989), "A Survey of Expert Systems in Manufacturing and Process Planning," *Computers in Industry*, 11, No. 2, pp. 195-204.

Hadavi, K., M.S. Shahrary and K. Voigt, "ReDS--A Dynamic Planning, Scheduling, and Control System for Manufacturing", *Journal of Manf. Systems*, 9, No. 4, pp. 332-244.

Hadavi, K., W. Hu, T. Chen and C. Lee (1992), "An Architecture for Real-Time Distributed Scheduling," *AI Magazine*, Fall, pp. 46-56.

Helferich, O.K., C.J. Espel and L.A. Taylor (1988), "Expert Systems: Logistics Applications in Support of Materials Planning and Production," *Expert Systems and Intelligent Manufacturing*, M.D. Oliff (ed.), Elsevier Science Pub. Co., p
154-173.

Hilliard, M.R., G.E. Liepins and M. Palmer (1988), "Machine Learning Applications to Job Shop Scheduling," *Proceedings of the First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Univ. of Tennessee Space Institute, June 1-3, ACM Press, pp. 205-211.

Hilliard, M.R., G.E. Liepins, M. Palmer, M. Morrow and J. Richardson (1987), "A Classifier-Based System for Discovering Scheduling Heuristics," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, J.J. Grefenstetts (ed.), Lawrence Erlbaum Pub., Hillsdale, N.J., pp. 231-235.

Hilliard, M.R., G.E. Liepins, G.Rangarajan and M. Palmer (1989), "Learning Decision Rules for Schedulin Problems: A Classifier Hybrid Approach," *Proceedings of the Sixth International Workshop on Machine Learning*, A. M. Segre (ed.), Cornell Univ., Morgan Kaufmann Pub., San Mateo, Calif., pp. 188-190.

Holland, J. H. (1975), *Adaption in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press.

Holland, J.H. (1985), "Properties of the Bucket Brigade Algorithm," *Proceedings of the First International Conference on Genetics and their Applications*, J.J. Grefenstette (ed.), Carnegie-Mellon Univ., Pittsburgh, pp. 1-7.

Holland, J. H. (1992), "Genetic Algorithms," *Scientific American*, July, pp. 66-72.

Hopfield, J.J and D.W. Tank (1985), "Neural Computation of Decisions in Optimization Problems," *Biol. Cybernetics*, Vol. 52, pp. 141-152.

Huang, D. (1989), "The Context-Array Bucket-Brigade Algorithm: An Enhanced Approach to Credit-Apportionment in Classifier Systems," *Proceedings of the Third International Conference on Genetic Algorithms*, D. Schaefer (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 311-316.

Husbands, P., F. Mill, and S. Warrington (1991), "Genetic Algorithms, Production Plan Optimization and Scheduling", *Parallel Problem Solving from Nature*, H. Schwefel and R. Manner (eds.), Lecture Notes in Computer Science, No. 496, Springer Verlag, Berlin, pp. 80-84.

Inoue, I and M. Fuyuki (1989), "A Know-How Activated Simulation Tool-System for Production Management Support," *Knowledge Based Production Management Systems*, J. Browne (ed.), North-Holland, Amsterdam, pp. 73-82.

Jain, S., K. Barber, and D. Osterfeld (1990), "Expert Simulation for On-Line Scheduling," *Communication of the ACM*, 33, No. 10, pp. 54-62.

Kanet, J.J., and H.H. Adelsberger (1987), "Expert Systems in Production Scheduling," *European Journal of Operational Research*, 29, pp. 51-59.

Kanet, J.J. and V. Sridharan (1990), "The Electronic Leitstand: A New Tool for Shop Scheduling," *Manufacturing Review*, 3, No.3, pp. 161-169.

Kemp, F.K. (1988), "Artificially Intelligent Tools for Manufacturing Process Planners," *Proceedings of the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, M.D. Oliff (ed.),Benjamin/ Cummings, San Francisco, pp. 131-163.

Kempf, K.G. (1985), "Manufacturing and Artificial Intelligence," *Robotics*, 1, No. 1, pp. 13-26.

Kerr, R.M., and R.V. Ebsary (1988), "Implementation of an Expert System for Production Scheduling," *European Journal of Operational Research*, 33, pp. 17-29.

Koton, P. (1989), "SMARTplan: A Case-Based Resource Allocation and Scheduling System," *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Pensacola Beach, Fla., K. Hammond (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 285-289.

Koton, P. (1989), "Evaluating Case-Based Problem Solving," *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Pensacola Beach, Fla., K. Hammond (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 173-175.

Kumar, P.R., and J.R. Perkins (1989), "Stable, Distributed, Real-Time Scheduling of Flexible Manufacturing Assembly/ Disassembly Systems," *IEEE Transactions on Automatic Control*, 34, No. 2, pp. 139-148.

Kumara, S., S. Joshi, C.L. Moodie, R.L. Kashyap, and T.C. Chang (1986), "Expert Systems in Industrial Engineering," *International Journal of Production Research*, 24, No. 5, pp. 1107-1126.

Kurzwiel (1990), *The Age of Intelligent Machines*, MIT, Cambridge, Mass.

Kusiak, A. (1987), "Designing Expert Systems for Scheduling Automated Manufacturing," *Industrial Engineering*, 27, pp. 42-46.

Kusiak A., and M. Chen (1988), "Expert Systems For Planning and Scheduling Manufacturing Systems," *European Journal of Operations Research*, 34, pp. 113-130.

Lamatsch, A., M. Morlock, K. Neumann, and T. Rubach (1988), "SCHEDULE-An Expert-like System for Machine Scheduling," *Annals of Operations Research*, 16, pp. 425-438.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (1993), "Sequencing and Scheduling: Algorithms and Complexity," to appear in *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory.*

Lawrence, S.R., and T.E. Morton (1986), "Patriarch: Hierarchical Production Scheduling," *Symposium on Real Time Optimization in Automated Manufacturing Facilities*, National Bureau of Standards, Gaithersburg Md., January, pp. 87-97.

Lecocq, P., and T. Guiot (1988), "An Expert Systems Application to Increase the Flexibility and Efficacy of Real-time FMS Controllers," *Expert Systems and Intelligent Manufacturing*, M.D. Oliff (ed.), Elsevier Science Publishing Company, Amsterdam, pp. 249-264.

Lee, J.K. and M.S. Suh (1988), "PAMS: A Domin-Specific Knowledge-Based Parallel Machine Scheduling System," *Expert Systems*, 5, No. 3, pp. 198-214.

LePape, C. and S.F. Smith (1987), "Management of Temporal Constraints for Factory Scheduling," Report No. CMU-RI-TR-87-13, June, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Levitt, R.E. (1990), "Knowledge Based Planning Systems: An Engineering Perspective," *First International Conference on Expert Planning Systems*, Brighton UK, June pp. 27-29.

Levitt, R.E. and J.C. Konz (1985), "A Knowledge-Based Systems for Updating Engineering Project Schedules," C.L. Dyn (ed.), *Applications of Knowledge-Based Systems to Engineering Analysis and Design*, ASME, New York, pp. 96-109.

Levy, A.B. and J.J. Wlassich (1988), "Knowledge-based Systems Applied to Advanced Materials Manufacturing," *Expert Systems and Intelligent Manufacturing: Proceedings of the Second International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, M.D. Oliff (ed.), Elsevier Science Pub., Amsterdam, pp. 138-153.

Liepins, G.E., M.R. Hilliard, M.Palmer and M. Morrow (1987), "Greedy Genetics," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, MIT, J.J.Grefenstette (ed.), Lawrence Erlbaum Asoc. Pub., Hillsdale, N.J., pp. 90-99.

Liepins, G.E., M.R. Hilliard, M. Palmer, and G. Rangarajan (1991), "Credit Assignment and Discovery in Classifier Systems," *International Journal of Intelligent Systems*, 6, pp. 55-69.

Liepins, G.E. and M.D. Vose (1990), "Representational Issues in Genetic Optimization," *Journal of Expt. Theor. Artif. Intell.*, 2, pp. 101-115.

Lin, G.Y. an J.J. Solberg (1992), "Integrated Shop Floor Control Using Autonomous Agents," *IIE Transactions*, 24, No. 3, pp. 57-71.

Liu, D. (1984), "Utilization of Artificial Intelligence in Manufacturing," *Proceedings of Autofact 6 Conference*, Detroit, October 1-4, pp. 2.60- 2.78.

Liu, D. (1986), "A Case Study of the HICLASS Software System, A Manufacturing Expert System," *Knowledge-Based Expert Systems for Manufacturing*, Winter Annual Meeting of the American Society of Mechanical Engineers, December, Pittsburgh, pp. 25-31.

Liu, D. (1989), "Expert Systems, AI, and Manufacturing," *Expert Systems: Planning/ Implementation Integration*, 1, No. 1, pp. 15-22.

Luger, G.F. and W.A. Stubblefield (1989), *Artificial Intelligence and the Design of Expert Systems*, Benjamin/Cummings Pub., San Francisco.

MacFarland, D.G. and F.H. Grant (1987), "Artificial Intelligence: Advances in Sequencing and Scheduling," *IIE Integrated Systems Conference Proceedings*, Pittsburgh, pp. 76-81.

Manivannan S. and J. Banks (1992), "Design of a Knowledge-Based On-Line Simulation System to Control a Manufacturing Shop Floor," *IIE Transactions*, 24, No. 3, pp 72-83.

Mark, W.S. (1989), "Case-Based Reasoning for Autoclave Management," *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Pensacola Beach, Fla., K. Hammond (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 176-180.

Martinez, J., P.R. Muro, M. Silva, S.F. Smith, and J.L. Villarroel (1988), "Merging Artificial Intelligence Techniques and Petri Nets for Real Time Scheduling and Control of Production Systems," *AI and Expert Systems in Scientific Computing*, of the IMACS Transactions on Scientific Computing-88 Series, ACM Press, New York, pp. 46-55.

May, J. H., L.G. Vargas, R.De, S.A. Slotnick, T.E. Morton, D.W. Pentico and G.J. Tatar (1991), "Multex: An Integrated AI/OR System For Factory Scheduling," *Proceedings of the Northeast DSI Conference*, Pittsburgh, April, pp. 67-72.

Mayer, J.R., D.T. Phillips, and R.E. Young (1987), "Artificial Intelligence Applications in Manufacturing," *Smart Manufacturing With Artificial Intelligence*, Computer and Automated Systems Association of SME, Dearbon, Mich., pp. 10-24.

Mayer, R.J. (1985), "Artificial Intelligence Applied to Manufacturing," *CIM Review,* 21, pp. 25-29.

Maza, M. (1989), "A Seagul Visits the Race Track," *Proceedings of the Third International Conference on Genetic Algorithms*, D. Schaefer (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 232-240.

McKay, K.N., J.A. Buzacott, N. Charness, and F.R. Safayeni (1991), "The Scheduler's Predictive Expertise-An Inter-disciplinary Perspective," *Artificial Intelligence in Operations Research*, North Holland, Amsterdam, pp. 64-76.

McKay, K.N., J.A. Buzacott and F.R. Safayeni (1989), "The Scheduler's Knowledge of Uncertainty: The Missing Link," *Knowledge Based Production Management Systems*, J. Browne (ed.), North-Holland, Amsterdam, pp. 171-189.

McKay,K.N., F.R. Safayeni and J.A. Buzacott (1988), "Job-Shop Scheduling Theory: What is Relevant?" *Interfaces*, 18, No. 4, pp. 84-90.

Miller, R.K. (1987), "Artificial Intelligence: A New Tool for Manufacturing," *Smart Manufacturing With Artificial Intelligence*, Computer and Automated Systems Association of SME, Dearbon, Mich., pp. 3-9.

Mohan, S., and D. Clancy (1989), "SIS-Rule Based Software for Automating Job Dispatch on the Factory Floor," *Proceedings from the Third International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, Pittsburgh, pp. 23-29.

Muhlenbein, H., M. Gorges-Schleuter and O. Kramer (1988), "Evolution Algorithms in Combinatorial Optimization," *Parallel Computing*, 7, pp. 65-88.

Muscettola, N., and S.F. Smith (1987), "A Probabilistic Framework for Resource-Constrained Multi-Agent Planning," *Proceedings, 10th International Joint Conference on Artifical Intelligence*, Milan, Italy, pp. 1063-1066.

Musen, M.A. (1989), "Automated Support for Building and Extending Expert Models," *Machine Learning*, 4, pp. 347-375.

Nakasura, S. and T. Yoshida (1992), "Dynamic Scheduling System Utilizing Machine Learning as a Knowledge Acquisition Tool," *Intl. Journal of Production Research*, 30, No. 2, pp. 411-431.

Niew, B.C., B.S. Lim and N.C. Ho (1990), "Knowledge Based Master Production Scheduler," *First International Conference on Expert Planning Systems*, Brighton, UK, June 27-29, pp. 88-93.

Nix, A. and M.D. Vose (1992), "Modeling Genetic Algorithms with Markov Chains," *Annals of Mathematics and Artificial Intelligence*, 5, pp. 79-88.

Noronha, S.J. and V.V.S. Sarma (1991), "Knowledge-Based Approaches for Scheduling Problems: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, 3, No. 2, pp. 160-171.

O'Grady, P. and K.H. Lee (1988), "An Intelligent Cell Control System for Automated Manufacturing," *International Journal of Production Research*, 26, No. 5, pp. 845-861.

O'Keefe, R. (1986), "Simulation and Expert Systems-A Taxonomy and Some Examples," *Simulation*, 46, No. 1, pp. 10-16.

Oliver, I.M., D.J. Smith and J.R.C. Holland (1987), "A Study of Permutation Crossover Operators on the Travelling Salesman Problem," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference*, J.J. Grefenstette (ed.), Lawrence Erlbaum Pub., Hillsdale, N.J., pp. 224-230.

Orciuch, E. and J. Frost (1984), "ISA: Intelligent Scheduling Assistant," *First Conference on Artificial Intelligence Applications*, IEEE Computer Society, Denver, December, pp. 314-320.

Ow, P.S. (1986), "Experiments in Knowledge-based Scheduling," Technical Report, Carnegie Mellon University, Pittsburgh.

Ow, P.S. and S.F. Smith (1986), "Towards an Opportunistic Scheduling System," *Nineteenth Annual Hawaii International Conference on System Sciences*, Honolulu, January, pp. 239-246.

Ow, P.S., S.F. Smith and R. Howie (1988), "A Cooperative Scheduling System," *Expert Systems and Intelligent Manufacturing*, M.D. Oliff (ed.), North-Holland, Amsterdam, pp. 70-89.

Pagallo, G. and D. Haussler (1989), "Two Algorithms that Learn DNF by Discovering Relevant Features," *Proceedings of the Sixth International Workshop on Machine Learning*, Morgan Kaufmann Pub., San Mateo, Calif, pp. 119-123.

Panwalkar, S.S. and W. Iskander (1977), "A Survey of Scheduling Rules," *Operations Research*, 25, pp. 45-61.

Parnas, D.L. (1988), "Why Engineers Should Not Use Artificial Intelligence," *Interfaces*, 26, No. 4, pp. 234-245.

Pargas, R.P. and J.C. Peck (1990), "Production Scheduling through Distributed Simulation," *Proceedings of the Fourth International Conference on Production and Operations Management*, Hilton Head Island, S.C., May, pp. 90-94.

Park, S.C., N. Raman and M.J. Shaw (1989), "Heuristic Learning in Pattern-Directed Scheduling," *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems*, Elsevier Science Pub., Amsterdam, pp. 369-376.

Parlar, M. (1989), "EXPIM: A Knowledge-Based Expert System for Production/ Inventory Modeling," *Int. J. Prod. Res.*, 27, No. 1, pp. 101-118.

Parunak, H.V.D. (1988), "Distributed AI Systems," *Artificial Intelligence: Implications for Computer Integrated Manufacturing*, A. Kusiak (ed.), IFS Publications, Kempston, UK, and Springer-Verlag, New York, pp. 225-251.

Perkins, J.R. and P.R. Kumar (1989), "Stable, Distributed Real-Time Scheduling of Flexible Manufacturing/Assembly/Disassembly Systems," *IEEE Transactions on Automatic Control*, 34, No. 2, pp. 139-148.

Piramuthu, S., S.C. Park, N. Raman, and M.J. Shaw (1991), "Integration of Simulation Modeling and Inductive Learning in an Adaptive Decision Support System," *Model Management Systems*, C. Bonczec and A.B. Whinston (ed.), IEEE Society Press, Los Altimos, Calif., pp. 71-80.

Powner, E.T. and D.H. Walburn (1990), "A Knowledge Based Scheduler," *First International Conference on Expert Planning Systems*, Brighton UK, June 27-29, pp. 82-87.

Preiss, K. and O. Shai (1989), "Process Planning by Logic Programming," *Robotics and Computer-Integrated Manufacturing*, 5, No. 1, pp. 1-10.

Quinlan, J.R. (1986), "Induction of Decision Trees," *Machine Learning*, 1, pp. 81-106.

Rabelo, L.C., S. Alptekin, and A.S. Kiran (1990), "Synergy of Artificial Neural Networks and Knowledge-Based Expert Systems for Intelligent FMS Scheduling," *Proceedings IEEE Intl. Joint Conference on Neural Networks*, IEEE Neural Networks Council, Ann Arbor, Mich., Vol. 1, pp. 359-366.

Ready, C.M., W.H. Simmonds, and J.C. Taunton (1990), "A Knowledge Based Planning and Scheduling Toolkit for the Process Industries," *First International Conference on Expert Planning Systems*, Brighton UK, June 27-29, pp. 110-113.

Realiff, M.J., and G. Stephanopoulas (1990), "Machine Learning for the Improvement of Scheduling Algorithms," MIT Industrial Liason Program Department LISPE-90-083, Cambridge, Mass.

Reinschmidt, K.F., J.H. Slate, and G.A. Finn (1990), "Expert Systems for Plant Scheduling using Linear Programming," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C., May, pp. 198-211.

Roboam, M. and M.S. Fox (1990), "Intelligent Networking: Towards Integrating the Manufacturing Enterprise," *Proceedings of the Fourth International Conference on Expert Systems on Production Operations Management*, Hilton Head Island, S.C., May, pp. 226-232.

Rodammer, F.A., and K.P. White (1988), "A Recent Survey of Production Scheduling," *IEEE Transactions on Systems, Man and Cybernetics*, 18, No. 6, pp. 841-851.

Riolo, R.L. (1987a), "Bucket Brigade Performance: I. Long Sequences of Classifiers," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms*, MIT, J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 184-195.

Riolo, R.L. (1987b), "Bucket Brigade Performance: II. Default Hierarchies," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms*, MIT, J.J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 196-201.

Robertson, G.G. and R.L. Riolo (1988), "A Tale of Two Classifier Systems," *Machine Learning*, 3, pp. 139-159.

Sadeh, N., and M.S. Fox (1990), "Variable and Value Ordering Heuristics for Activity-Based Job-Shop Scheduling," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C., May, pp. 276-283.

Sadeh, N. and M.S. Fox (1989), "Preference Propagation in Temporal/Capacity Constraint Graphics," Paper No. CMU-RI-TR-89-2, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Sarin, S.C., and R. Salgame (1989), "A Knowledge-Based System Approach to Dynamic Scheduling," *Knowledge-Based Systems in Manufacturing*, A. Kusiak (ed.), Taylor and Francis, New York, pp. 173-206.

Savell, D.V., R.A. Perez, and S.W. Koh (1989), "Scheduling Semiconductor Wafer Production: An Expert System Implementation," *IEEE Expert*, Fall, pp. 9-15.

Schaffer, J.D. (1987), "Some Effects of Selection Procedures on Hyperplane Sampling by Genetic Algorithms," *Genetic Algorithms and Simulated Annealing*, L. Davis (ed.), Morgan Kauffman Pub., San Mateo, Calif., pp. 36-40.

Schraudolph, N.N. and R.K. Belew (1991), "Dynamic Parameter Encoding for Genetic Algorithms," Tech. Report CS 90-175, Univ. of California, San Diego.

Shalin, V.L. (1990), "A Formal Analysis of Machine Learning Systems for Knowledge Acquisition," *Knowledge Based Systems Vol. 3: Machine Learning and Uncertain Reasoning*, B. Gaines and J. Boose (eds.), Kluwer Academic Press, Netherlands, pp. 46-59.

Shannon, R.E. (1984), "Artificial Intelligence and Simulation," *Proceedings of the 1984 Winter Simulation Conference*, S. Shepard, U.W. Pooch and C.D. Pegden (eds.), Dallas, November, North Holland, Netherlands, pp. 3-9.

Shannon, R.E. (1991), "Knowledge Based Simulation Techniques for Manufacturing," *Intl. J. Prod. Res.*, 26, No. 5, pp. 953-973.

Shaw, M.J. (1988a), "FMS Scheduling as Cooperative Problem Solving," *Operations Research*, 17, pp. 323-346.

Shaw, M.J. (1988b), "Dynamic Scheduling in Cellular Manufacturing Systems: A Framework for Networked Decision Making," *Journal of Manufacturing Systems*, 7, No. 2, pp. 83-94.

Shaw, M.J. (1988c), "Knowledge-based Scheduling in Flexible Manufacturing System: An Integration of Pattern-directed Inference and Heuristic Search," *International Journal of Production Research*, 26, No. 5, pp. 821-844.

Shaw, M.J. (1989), "A Pattern-Directed Approach to Flexible Manufacturing: A Framework for Intelligent Scheduling, Learning, and Control," *International Journal of Flexible Manufacturing Systems*, 2, pp. 121-144.

Shaw, M.J., N. Raman and S.C. Park (1991), "Intelligent Scheduling with Machine Learning Capabilities: The Induction of Scheduling Knowledge," Technical Report AI-DSS-91-01, Beckman Institute, University of Illinois, Urbana-Champaign, 1991, forthcoming, *IEE Transactions.*

Shaw, M.J., J.J. Solberg and T.C. Woo (1992), "System Integration in Intelligent Manufacturing: An Introduction", *IIE Transactions*, Vol. 24, No. 3, pp. 1-7.

Shaw, M.J., P.L. Tu, and P. De (1989), "Applying Machine Learning to Model Management in Decision Support Systems," *Decision Support System*, 4, pp. 285-305.

Shaw, M.J. and A.B. Whinston (1986), "Application of Artificial Intelligence to Planning and Scheduling in Flexible Manufacturing," *Flexible Manufacturing Systems: Methods and Studies*, A. Kusiak (ed.), North Holland, Amsterdam, Netherlands, pp. 223-242.

Shaw, M.J. and A.B. Whinston (1988), "A Distributed Knowledge-Based Approach to Flexible Automation: The Contract Net Framework," *International Journal of Flexible Manufacturing Systems*, 1, pp. 85-104.

Shaw, M.J. and A.B. Whinston (1989), "An Artificial Intelligence Approach to the Scheduling of Flexible Manufacturing Systems," *IIE Transactions*, 21, No. 2, pp. 170-183.

Shen, S. and Y. Chang (1988), "Schedule Generation in Flexible Manufacturing System: A Knowledge-Based Approach," *Decision Support Systems*, 4, pp. 157-166.

Sidhu, S., S. Gupta and F. Vlach (1988), "Issues in the Design and Implementation of Intelligent Scheduling Systems," *Proceedings from the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, M.D. Oliff (ed.), Benjamin/Cummings, San Francisco, pp. 375-379.

Simon, H.A. (1987), "Two Heads Are Better Than One: The Collaboration between AI and OR," *Interfaces*, 17, No. 4, pp. 8-15.

Smith, S.F. (1983), "Exploiting Temporal Knowledge to Organize Constraints," Report No. CMU-RI-TR-83-12, Intelligent Systems Laboratory, The Robotics Institute, Carnegie Mellon University, Pittsburgh.

Smith, S.F. (1987), "A Constraint Based Framework for Reactive Management of Factory Schedules," *Proceedings International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, Charleston S.C., May, M.D. Oliff (ed.), Benjamin Cummings, San Francisco, pp. 113-130.

Smith, S.F. (1988a), "A Constraint-Based Framework for Reactive Management of Factory Schedules," *Proceedings of the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, M.D. Oliff (ed.), Benjamin/Cummings, San Francisco, pp. 113-130.

Smith, S.F. (1988b), "Knowledge-Based Scheduling Systems," *Proceedings of the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, M.D. Oliff (ed.), Benjamin/Cummings, San Francisco, pp. 381-383.

Smith, S.F. (1989), "The OPIS Framework for Modeling Manufacturing Systems," Report No. CMU-RI-TR-89-30, Carnegie Mellon University, Pittsburgh.

Smith, S.F. (1991), "Knowldge-Based Production Management: Approaches, Results and Prospects", Report No. CMU-RI-TR-91-21, Carnegie Mellon University, Pittsburgh.

Smith, S.F., M.S. Fox, and P.S. Ow (1986), "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems," *AI Magazine*, Fall, pp. 45-61.

Smith, S.F., and J.E. Hynynen (1987), "Integrated Decentralization of Production Management: An Approach for Factory Scheduling," *Proceedings 1987 Symposium on Integrated and Intelligent Manufacturing*, ASME Annual Winter Conference, Boston, December, pp. 136-144.

Smith, S.F. and P.S. Ow (1985), "The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks," *Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, August, pp. 1013-1015.

Smith, S.F., P.S. Ow, C. Lepape, B. Mclaren, and N. Muscettola (1986), "Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans," *Proceedings 1986 SME Conference on AI in Manufacturing*, Long Beach, Calif., pp. 273-281.

Smith, S.F., P.S. Ow, N. Muscettola, J-Y. Potvin, and D.C. Matthys (1990), "An Integrated Framework for Generating and Revising Factory Schedules," *Journal of the Operational Research Society*, 41, No. 6, pp. 539-552.

Smith, S.F., P.S. Ow, and J-Y. Potvin (1990), "OPIS: An Opportunistic Factory Scheduling System," *The Third International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, IEA/AIE 90, Charleston, S.C., July 15-18, pp. 169-174.

Spears, W.M. and K. DeJong (1991), "An Analysis of Multi-Point Crossover," *Foundations of Genetic Algorithms*, Gregory J.E. Rawlins (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 301-315.

Steffen, M.S. (1986), "A Survey of Artificial Intelligence-Based Scheduling Systems," *Proceedings Fall Industrial Engineering Conference*, Pittsburgh, December 7-10, pp. 395-405.

Steffen, M.S., and T.J. Greene (1986a), "Hierarchies of Sub-Periods in Constraint-Directed Scheduling," *Symposium on Real Time Optimization in Automated Manufacturing Facilities*, National Bureau of Standards, Gaithhersburg, Md., January, pp. 167-183.

Steffen, M.S., and T.J. Greene (1986b), "A Prototype System for Scheduling Parallel Processors Using Artificial Intelligence Methods," *Annual International Industrial Engineering Conference Proceedings*, Pittsburgh, pp. 156-164.

Steffen, M.S., and T.J. Greene (1987), "Automating the Scheduling of Parallel Machines," *Smart Manufacturing with Artificial Intelligence*, Computer and Automated Systems Association of SME, Dearbon, Mich. pp. 119-134.

Suh, J,Y. and D. Van Gucht (1987), "Incorporating Heuristic Information into Genetic Search," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorthms*, MIT, J. Grefenstette (ed.), Lawrence Erlbaum Assoc. Pub., Hillsdale, N.J., pp. 100-107.

Sullivan, G., and K. Fordyce (1990), "IBM Burlington's Logistics Management System," *Interfaces*, 20, No. 1, pp. 43-64.

Sycara, K., S. Rothm, N. Sadeh, and M. Fox (1990), "An Investigation into Distributed Constraint-directed Factory Scheduling," *Sixth IEEE Conference on AI Applications*, Santa Barbara, Calif., March, pp. 281-288.

Syswerda, G. (1989), "Uniform Crossover in Genetic Algorithms," *Proceedings of the Third International Conference on Genetic Algorithms*, D. Schaefer (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 2-9.

Syswerda, G. (1991)," Schedule Optimization Using Genetic Algorithms," *Handbook of Genetic Algorithms*, L. Davis (ed.), Ch. 13, Van Nostrand Reinhold Pub., New York, pp. 333-349.

Thesen A., and L. Lei (1986), "An Expert System for Scheduling Robot in a Flexible Electroplating System with Dynamically Changing Work Loads," *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models andApplications*, K.E. Stecke (ed.), Elsevier Science Pub., Amsterdam, pp. 555-566.

Tsai, L. and G.J. Koehler (1993), "The Accuracy of Concepts Learned from Induction," *Decision Support Systems*, forthcoming.

Turban, E. (1990), *Decision Support and Expert Systems: Management Support Systems*, 2nd edition, 1990, Macmillan Pub. Co., New York.

Vose, M.D. (1993), "Models of Genetic Algorithms," Working Paper, University of Tennessee, Knoxville.

Vose, M.D. and G.E. Liepins (1990), "Schemata Disruption," *Proceedings of the Fourth International Conference on Genetic Algorithms*, Univ. of California, San Diego, July, Morgan Kaufmann Pub., San Mateo , Calif., pp. 237-242.

Vose, M.D. and G.E. Liepins (1991), "Punctuated Equilibria in Genetic Search," *Complex Systems*, 5, pp. 31-44.

Whitley, D. (1989), "The Genitor Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best," *Proceedings of the Third International Conference on Genetic Algorithms*, D. Schaefer (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 116-121.

Whitley, L.D. (1991), "Fundamental Principles of Deception in Genetic Search," *Foundations of Genetic Algorithms*, Gregory J.E. Rawlins (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 221-241.

Whitley, L.D., T. Starkweather and D. Fuquay (1989), "Scheduling Problems and the Travelling Salesman: the Genetic Edge Recombination Operator," *Proceedings of the Third International Conference on Genetic Algorithms*, D. Schaefer (ed.), Morgan Kaufmann Pub., San mateo, Calif., pp.133-140.

Whitley, L.D., T. Starkweather and D. Shaner (1991), "The Travelling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination," *Handbook of Genetic Algorithms*, L. Davis (ed.), Ch. 13, Van Nostrand Reinhold Pub., New York, pp. 351-372.

Wilson, S.W. and D.E. Goldberg (1989), "A Critical Review of Classifier Systems," *Proceedings of the Third International Conference on Genetic Algorithms*, D. Schaefer (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 86-91.

Wilson, S.W. (1987), "Hierarchical Credit Allocation in a Classifier System," *Genetic Algorithms and Simulated Annealing*, L. Davis (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 136-149.

Wright, A.H. (1991), "Genetic Algorithms for Real Parameter Optimization," *Foundations of Genetic Algorithms*, Gregory J.E. Rawlins (ed.), Morgan Kaufmann Pub., San Mateo, Calif., pp. 205-218.

Wysk, R.A., S.Y.D. Wu, and N.S. Yang (1986), "A Multi-Pass Expert Control System (MPECS) for Flexible Manufacturing Systems," *Symposium on Real-Time Optimization in Automatic Manufacturing Facilities*, National Bureau of Standards, Gaithersburg, Md., January, pp. 9-25.

Yih, Y. (1990a), "Trace Driven Knowledge Acquisition for Rule Based real Time Scheduling Systems," *Journal of Intelligent Manufacturing*, 1, pp. 217-230.

Yih, Y. (1990b), "Learning Decision Rules for FMS from the Optimal Policy of User-Based Semi-Markov Decision Processes," *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C., pp. 175-183.

Yih, Y. (1992), "Learning Real Time Scheduling Rules from Optimal Policy of Semi-Markov Decision Processes," *International Journal of Computer Integrated Manufacturing*, 5, No. 3, pp. 171-181.

Yih, Y., T.P. Liang and H. Moskowitz (1992), "Robot Scheduling in a Circuit Board Production Line--A Hybrid OR/ANN Approach," Working Paper, Purdue University, Indiana.

Yih, Y. and A. Theisen (1991), "Semi-Markov Decision Models for Real-Time Scheduling," *International Journal of Production Research*, 29, No. 11, pp. 2331-2346.

Young, R.E., and M.A. Rossi (1988), "Toward Knowledge-Based Control of Flexible Manufacturing Systems," *IIE Transactions*, 20, No. 1, pp. 36-43.

Zahedi, F., "Artificial Intelligence and the Management Science Practitioner: The Economics of Expert Systems and the Contribution of MS/OR," *Interfaces*, 17, No. 5, pp. 72-81.

Zhou, D.N., V. Cherkassky, T.R. Baldwin and D.W. Hong (1990), "Scaling Neural Network for Job-Shop Scheduling," Proceedings IEEE Intl. Joint Conference on Neural Networks, San Diego, IEEE Neural Networks Council, Ann Arbor, Mich., Vol. 3, pp. 46-52.

Zhou, H.H. (1990), "CSM: A Computational Model of Cumulative Learning", *Machine Learning*, 3, pp. 382-406.

BIOGRAPHICAL SKETCH

Siddhartha Bhattacharyya was born in Shillong, India, in July, 1964. He obtained his bachelors degree in Electronics and Communications Engineering from the Birla Institute of Technology, Mesra, India, in 1986. He subsequently worked as Customer Support Engineer for three years with a leading computer manufacturer in India.

In 1989, Mr. Bhattacharyya came to the University of Florida and enrolled in the doctoral program in Information Systems. At the University of Florida, he has been a recipient of the Grinter Graduate Fellowship, and is a member of the Beta Gamma Sigma and Alpha Iota Delta academic honor societies. On completion of his doctorate, he will be taking up a career in university research and teaching.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Gary J. Koehler, Chairman
Professor of Decision and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Chung Yee Lee
Associate Professor of
Industrial and Systems
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Richard A. Elnićki
Professor of Decision and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Anthal Majthay
Associate Professor of
Decision and Information
Sciences

This dissertation was submitted to the Graduate Faculty of the Department of Decision and Information Sciences in the College of Business and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1993

Gene Hemp

Dean, Graduate School